Universidad Autónoma de Madrid
Escuela Politécnica Superior / Facultad de Ciencias
Departamento de Ingeniería Informática / Departamento de Matemáticas

SVRs and Uncertainty Estimates

Master's thesis presented to apply for the
Master's double–degree of Investigation and Innovation in Information and Communications Technology + Master of Applied Mathematics

By
Jesús Prada Alonso

under the direction of
José R. Dorronsoro Ibero and Eugenio Hernández Rodríguez

Madrid, July 9, 2015

Contents

Abstract

While Support Vector Regression, SVR, is one of the algorithms of choice in modeling problems, construction of its error intervals seems to have received less attention. In addition, general noise cost functions for SVR have been recently proposed and proved to be more effective when a noise distribution that fits the data is properly chosen.

Taking these two factors into account, this thesis has five main goals: First, describe a direct approach to build error intervals, based on the assumption of residuals following some probability distribution, how to fit these noise models and how to estimate their parameters. Second, give a comparison between intervals resulting from this method, with and without prior use of k-means and other methods to split the data, versus intervals built using a Bayesian approach. Third, study which distribution assumption provides better intervals in real-world problems, such as wind and solar energy forecasting, medical regression problems and prediction of sporting events outcome. Fourth, analyze if intervals result of this approach can be used as a detector of the distribution of errors, and hence become a good way to choose the noise assumption in a general noise model. And finally, give explicit formulations for SVR models using loss functions corresponding to different assumptions of noise distribution in the data, such as the Laplace or the Weibull distirbution.

Acknowledgements

This thesis has been developed with partial support from Spain's grants TIN2013-42351-P and S2013/ICE-2845 CASI-CAM-CM and also of the Cátedra UAM–ADIC in Data Science and Machine Learning. I also want to gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM and Red Eléctrica de España for kindly supplying wind and solar energy data.

The research presented here would not have been possible without the support and guidance of the members of the Machine Learning Group from UAM, with a special mention to my tutor José, who helped me to find my path in the long, complex and often hard, but extremely rewarding, life of a researcher.

I want to also thank my professors and, even more, my colleagues during my time at the university, the Instituto de Ingeniería del Conocimiento, IIC, and the Machine Learning Group from UAM, who help to ignite the spark of curiosity and thirst for knowledge, specially in this fascinating world of machine learning and big data, in me. Professor Juan Luis Vázquez and my colleagues Sara and Luis are some of the most important names here, but surely others have also played his part in what I am today.

Last but not least, special thanks to Yvonne, my closest support during these months of hard work and dedication, always there in the good and bad moments, and whose previous experience and knowledge about energy prediction have suppose an invaluable assistance to this thesis.

Chapter 1

Introduction

Extracting useful information from data has been a vital task in many studies and applications, both academic and corporate, for many years. The first expert systems, computer systems that emulates the decision-making ability of a human expert [1], appeared in the 1970s and clearly dominated the field of artificial intelligence during the 1980s. In these systems, the decision-making algorithm is explicitly coded, primarily as if-then rules. In contrast to expert systems, machine learning [2], ML, try to infer from the data the best algorithm to model the problem and give a solution to it, whether it is a division into clusters, a classification into previously specified groups or the prediction of a real number. In ML the dependence on and need of expert knowledge is lessened, although not completely removed. Furthermore, more general and less ad hoc applications can be built using ML models. These and others advantages have made machine learning a very popular tool and one that has been widely studied and used in a variety of problems in recent years. Its popularity has been strengthened by the recent rise in importance of big data problems.

1.1 Machine Learning

Machine learning is a scientific discipline that aims to build computer systems that automatically improve with experience, exploring the construction and study of algorithms that can learn from data.

This field covers a broad range of learning tasks, such as:

- Design of unmanned aerial vehicles, UAVs, that learn to navigate and interact with other UAVs from their own experience.
- How to use historical medical records to learn which people could suffer from some particular disease, such as alzheimer or parkinson, or which patients will respond best to which treatments.
- Recommender systems, or how to build search engines that automatically customize to their users interests using their information and past actions.
- Business intelligence, BI, that aims to transform data from companies into meaningful and useful information for business analysis purposes.
- Forecasting of wind, solar or other type of energy production in different geographic areas learning from past productions and weather information.

- Prediction of sporting events outcome using past results, statistics and information from other sources such as social networks.

In [2] this definition of learning is given: *"A machine learns with respect to a particular task T, performance metric P, and type of experience E, if the system reliably improves its performance P at task T, following experience E"*. Using wind energy prediction as an example, wind energy production forecast using weather information would be task T, past data of weather and energy production would form experience E and the performance P of the system would be measured using a particular metric, such as the mean absolute error, MAE, or the mean squared error, MSE, defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\hat{f}(X_i) - y_i| \tag{1.1.1}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (\hat{f}(X_i) - y_i)^2 \ , \tag{1.1.2}$$

where $\{X_i\}_{i=1}^{N} = \{(x_1, x_2, ..., x_d)^T\}_{i=1}^{N}$, is the weather information used as input data, $\hat{f}(X)$ is the forecast outputted by the machine learning model given input x, $\{y_i\}_{i=1}^{N}$ is the real wind energy production or *target*, N is the number of *instances* or samples and d is the number of variables, or *dimensions*, in the data.

As said earlier, ML encompasses a wide variety of problems. In [3] and [4] these tasks are divided into three main groups:

1. **Supervised Learning**: In supervised learning, a labeled **training dataset** is used to train the model. **Labels** can be categorical, representing the belonging of a particular instance to a specific class, being 0 and 1 standard labels for a 2-class problem, or a discrete label in the form of a real number. In the first case, the task is called a **classification problem** and in the second case a **regression problem**. In both cases, the principles of the machine learning cycle are the same, but algorithms and **objective functions** used are different for each type of problem. Parameters of the model are chosed to minimize a particular objective function for this training set.

   The model built from this training process is then used to predict the class or value of new instances belonging to a labeled **test dataset**. Prediction errors result of this process are used as a measure of the model accuracy.

   Additionaly, a third set of labeled data, called **validation dataset**, can be used to select the best hyperparameters for the ML model. Frequently, this third dataset is not used and, instead, a technique called **cross-validation**, CV, [5] is employed to find the optimal hyperparameters. When this method is used, the training dataset is divided into k subsets, then the model is trained using $k-1$ of these subsets and the remaining one is used as validation set. This process is repeated k times until all subsets have performed the role of validation set. The errors result of these k iterations are then averaged and used as validation error. In both cases, fixed validation dataset or CV, hyperparameters chosen are the ones that minimize this validation error.

One of the key factors in supervised learning models is the **bias-variance tradeoff** to avoid *overfitting* and *underfitting* of the model to the data. In the first case, the model has too high variance, adjusting too much to the variations in the data set. In the second case, the model has too much bias, remaining mainly unaffected by changes in the input data. To deal with this, normally a **regularization or penalty term** is added to the usual error measure or **loss function** to form the objective function that will be minimized to build the model. One example is *ridge regression* [6], a model that has as objective function

$$\frac{1}{N}\sum_{i=1}^{N}(X_i^T\beta + \beta_0 - y_i)^2 + \frac{\lambda}{2}||\beta||^2 \ , \tag{1.1.3}$$

where $\hat{f}(X_i) = X_i\beta + \beta_0$ is the output of the model, β_0 is a constant called the *bias* of the model and $\beta = (\beta_1, \beta_2, ..., \beta_d)^T$ are the parameters or coefficients of the model.

2. **Unsupervised Learning**: In unsupervised learning there is **no labels** in the data and the goal is to find hidden structure in this unlabeled data, whether it is to **cluster data** into different groups, **estimate the distribution** of the data or build **latent variable models**.

   In this type of tasks, there is **no error or reward** signal to evaluate a potential solution. Sometimes, unsupervised learning techniques are employed prior to applying supervised learning models as a data pre-processing step. In this case, goodness of unsupervised learning methods can be measured by their impact on the accuracy of the corresponding supervised learning model.

3. **Reinforcement Learning**: In this type of tasks the concern is the problem of finding suitable actions to take in a given situation in order to maximize a **reward**. Here the learning model is **not given labels of optimal outputs**, in contrast to supervised learning, but must instead discover them using an iterative process of **trial and error**. Usually, there is a sequence of states and actions in which the model is interacting with its environment, and frequently the current action not only has an impact on the immediate reward but also affects the reward at all subsequent time steps. Only at the end of this process the reward signal, positive or negative, is received.

   Finding a balance between *exploration*, in which the system tries out new kinds of actions to see how effective they are, and *exploitation*, in which the system makes use of actions that are known to yield a high reward, is vital in these learning algorithms. This is called the **exploration-exploitation trade-off**.

This thesis focuses its attention on supervised learning problems, although some unsupervised learning techniques are applied as a preprocessing step to divide data into clusters.

Usually, the design of a supervised learning system entails a iterative cycle composed of several steps, where some of these steps are carried out again in each iteration. Although different divisions of this process into steps have been described, normally there are five stages:

1. **Data collection**: Consists in gathering the data needed to train, validate and test the model. Sometimes it is a very costly stage, so a tradeoff between the volume of data collected and the cost of this gathering process must be made. It is also important to keep in mind the complexity of the problem and of the model chosen to decide when we have an adequately large amount of data for a particular problem.

2. **Feature choice**: There are two different methods to carry out feature choice. *Feature selection* pick a subset of the original features and discard the rest, while *feature extraction* generates derived variables from the original ones. Both techniques achieve a dimensionality reduction of the data, where features selected are intended to be informative and non redundant.

   Dimensionality reduction can be made to improve the accuracy of the ML model, reducing the risk of overfitting, although for complex models with a regularization term this is not really necessary and feature choice is done implicitly by the model. In addition, dimensionality reduction can also provide a significant decrease in computational cost, vital in many real-world ML systems, particularly in those that need to be able to give a real time response, such as fraud detection in banking systems.

3. **Model choice**: There is a wide variety of ML models for supervised learning problems, ranging from a simple linear regression to deep learning techniques such as deep neural networks.

   The optimal model for a particular problem depends on factors as the nature of the problem and its complexity, data available and its dimensions and the presence of noise in the data. These factors must be taking into account when choosing the model to use and the common mistake of choosing a particular model for some personal preference and not for being the most suitable for the problem should be avoided when opting for a model to solve a real-world problem.

4. **Train and validation**: This step comprise the use of the training and validation datasets as described earlier to choose the optimal parameters and hyperparameters, respectively, for the model selected in the previous step.

5. **Evaluation**: Once training and validation of the model is done, its performance over the test dataset is evaluated through some particular accuracy measure. Typical choices for regression problems are MAE (1.1.1) and MSE (1.1.2), and for classification problems the following ones are common measures:

$$Precision = \frac{TP}{TP + FP} \tag{1.1.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.1.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{1.1.6}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \ , \tag{1.1.7}$$

where TP are true positives, i.e. instances classified as positive by the model that are in fact positive, FP are false positives, instances classified as positive that are in fact negative, TN are true negatives, instances classified as negative that are in fact negative and FN are false negatives, instances classified as negative that are in fact positive.

Precision vs recall and ROC or FPR vs recall curves are also frequently used as evaluation measures for classification problems.

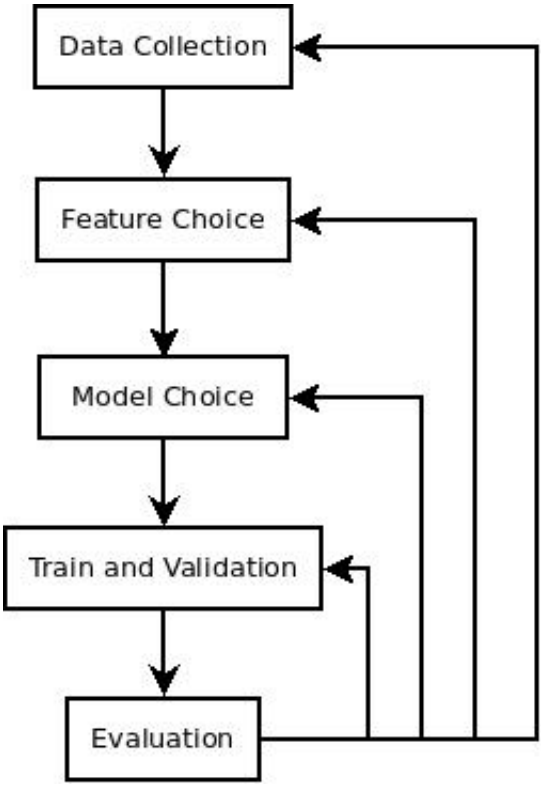


Figure 1.1.1: Design cycle for supervised learning systems.

1.2 Big Data

Big data is a rather new concept but one that has grow in importance very quickly in recent years in the field of computer science, swiftly becoming a key concept in many studies and applications. Despite its name, big data is not only related to the volume of raw information. For years, there has not exist a formal and consensual definition of what should be considered big data and what not. In 2012, the American firm Gartnet gave this definition [7]:

1.1. *Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization.*

A more recent and consensual definition of big data [8] states that

1.2. *Big Data represents the information assets characterized by such a high Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value*

Both definitions mention the three main properties of a big data problem, called the **3Vs** [9], that are:

1. **Volume**: Refers to the quantity of data in the problem. It is very important in this context and the name big data itself contains a term which is related to size.

2. **Variety**: The second of the 3 Vs of big data is the variety of the data. It makes reference to the diversity of the data and variance among the sources where it is collected. Usually, raw data is unstructured or has different structures depending on its source, so an appropiate pre-processing step is key.

3. **Velocity**: The term *velocity* in this context refers to the speed of generation of data and how fast the data must be processed to meet the demands and challenges of a particular task.

Other additional Vs have been proposed in recent years. An article from 2013 by Mark van Rijmenam adds four more, reaching a total of **7Vs**. Apart from the three mentioned above, these are:

1. **Variability**: Refers to data whose meaning is changeable. This is particularly the case when gathering data relies on language processing. Words do not have static definitions, and their meaning can vary wildly depending on context. Thus, programmes which can process context and decode the precise meaning of words through it have to be developed.

2. **Veracity**: Data is virtually worthless if it is not accurate, and in very seldom cases data available is noise free. Hence, it is again vital a good pre-processing stage that takes into account the usually noisy nature of data and produces a sufficiently accurate dataset before analysis can start.

3. **Visualization**: Once it has been processed and analyzed, you need a way of presenting the data in a manner that is readable and accessible, and this is what visualization refers to. Visualizations can contain tens, hundreds of even thousands of samples and variables, and finding a way to present this information that makes the findings clear is one of the challenges of big data.

4. **Value**: The potential value of big data is huge. However, the cost generated by the use of poor data is also really significant. In essence, data on its own is virtually worthless, the value of big data lies in rigorous analysis of accurate data, and the information and insights this provides to company and academic researchers.

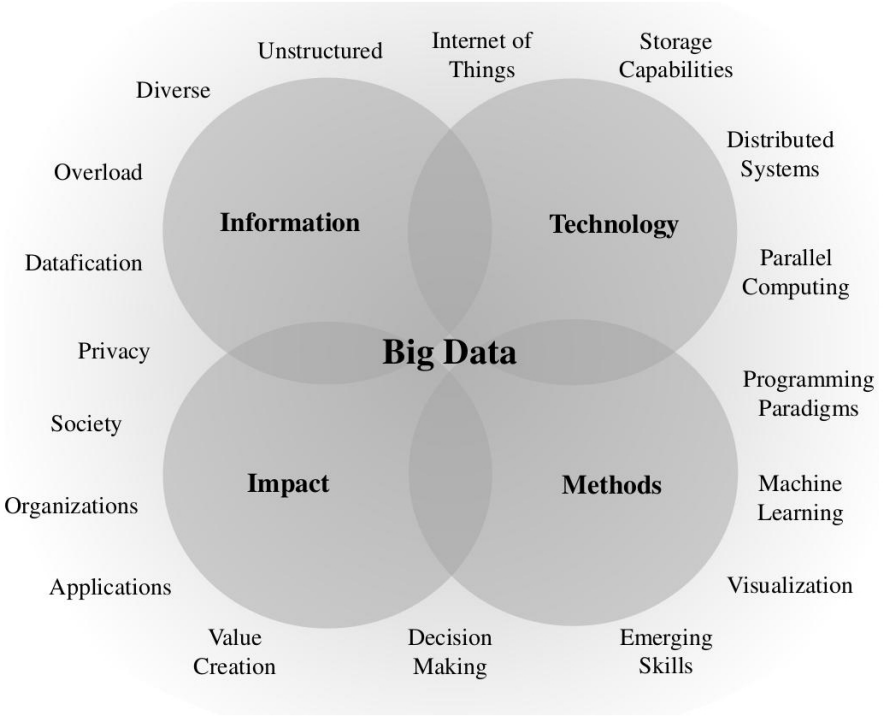


Figure 1.2.1: Big data main topics in existing research. Image from [8].

1.3 Machine Learning and Big Data. The Perfect Marriage

Machine learning and big data are two concepts that have extremely close ties, both benefiting from one another.

On the one hand, the potential of ML models to give accurate outputs is boosted with bigger (**volume**) and more diverse (**variety**) available datasets to train and validate them, providing that the data is accurate and meaningful (**veracity** and **value**). The raise of big data in recent years has provided ML scientists with more information to insert into their models and hence their potential have been enhanced very significantly.

On the other hand, big data problems have changed the entire way of thinking about knowledge extraction and interpretation. Traditionally, data science has always been dominated by trial-and-error analysis, an approach that becomes impossible when datasets are large and heterogeneous as occurs when big data comes into play. Ironically, availability of more data usually leads to fewer options in constructing predictive models, because very few tools allow for processing large datasets in a reasonable amount of time. In addition, traditional statistical solutions typically focus on static analytics that is limited to the analysis of samples that are frozen in time, which often results in surpassed and unreliable conclusions. Machine learning techniques allow researchers to overcome those problems and to build systems that can provide **online learning**, models updated after the arrival of new datapoints. ML models can also be used to build **real-time systems**, programs that must guarantee response within specified time constraints (**velocity**).

To help researchers to combine the use of ML and big data, several software have been developed in recent years. In addition to supercomputers and RPC architectures, the recent appearance of open frameworks that make easier computations for big data ML problems, such as *Apache Hadoop* [10], *Cloudera* [11], *Apache Mahout* [12] and *Apache Spark* [13], has boosted investigation in this line of research.

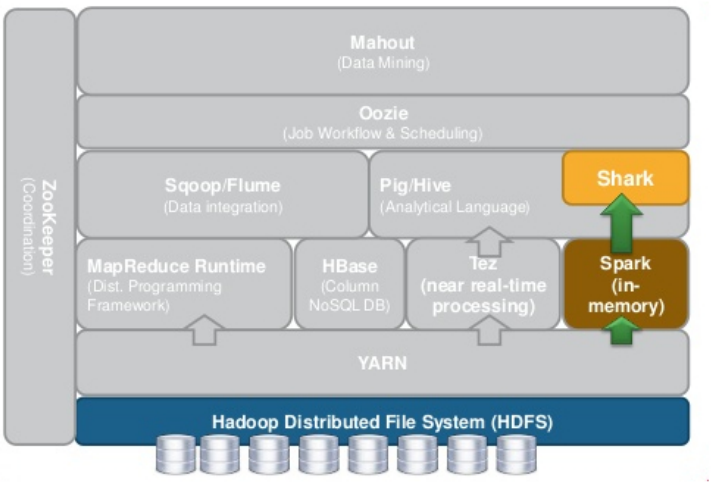


Figure 1.3.1: The Hadoop Ecosystem.

Also other non-open software, such as *SAS High-Performance Analytics*, make big data models for machine learning less computationally expensive, allowing analysts to try different and more complex options to deal with the problems in hand.

Chapter 2

Support Vector Machines

Support vector machines, SVMs, are one of the most powerful and popular techniques among machine learning models. It has been proved to perform well in real-world situations, giving excellent results in a wide variety of problems, such as stock market [14], wind energy [15] or solar radiation [16] forecasting.

This chapter starts with an explanation of the classical formulations for SVM in Section 2.1, both the ϵ-SVM version and the ν-SVM one. Sequential minimal optimization, or SMO, an algorithm for solving the quadratic programming problem that arises during the training of support vector machines, is then described in 2.2. Then, a Bayesian framework for SVR models is detailed in Section 2.3. Finally, we present in Section 2.4 a general noise version of the SVM for regression, or SVR, where a particular noise distribution for the data is assumed and plugged into the model.

2.1 Classical SVM

Here, an in-depth description of the classical formulation for SVM is given. First, we detailed the classification problem, which can be divided in two cases: a linear separable case and a linear non-separable case. Then, the section deals with the problem of non-linearity, introducing the key concept of the **kernel trick**. Afterwards, the regression problem is described. This case encompasses two versions of the same optimization problem, usually called ϵ-SVR and ν-SVR. For the first one, an explanation is given, both for the L1 and L2 versions. For the second one, a less thorough description is given, focusing on the non-linear case for regression tasks.

2.1.1 L1-ϵ-SVM. Classification

This section focus on the 2-class problem. For tasks with more classes, one SVM for each class, which will decide what are the samples of the dataset that belong to that particular class, have to be build.

Our training data consists of N pairs $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, with $x_i = (x_{i1}, x_{i2}, ..., x_ip)^T \in R^p$ the variables or input data and $y_i \in \{-1, 1\}$ the labels of the classes or *target*.

Define a hyperplane, H, by

$$H = \{x : f(x) = x_i^T \beta + \beta_0 = 0\} \ , \tag{2.1.1}$$

where $\beta = (\beta_1, \beta_2, ..., \beta_p)^T$ is a unit vector, $||\beta|| = 1$, representing the parameters of the model, and β_0 is the bias of the model.

$f(x)$ induces a classification rule given by

$$G(x) = sign(x_i^T \beta + \beta_0) \ . \tag{2.1.2}$$

The aim of Support Vector Classification is to obtain the best separating hyperplane possible. Three different cases are possible:

1. Linear Separable Case. Hard Margin Classification

This case seldom appears in real problems, although its explanation is interesting to introduce the SVM principles.

Since the classes are separable, we can find a function $f(x) = x_i^T \beta + \beta_0$ with $y_i f(x_i) > 0, \ \forall i$. Hence, we are able to find the hyperplane that creates the biggest margin between the training points for class 1 and -1 (see Figure 2.1.1). This is computed solving the following optimization problem

$$\begin{aligned} \max_{\beta, \beta_0} \quad & M \\ \text{subject to} \quad & y_i(x_i^T \beta + \beta_0) \geq M, \ i = 1, ...N \\ & ||\beta|| = 1 \ . \end{aligned} \tag{2.1.3}$$

As stated in [17] we can get rid of the $||\beta|| = 1$ constraint by replacing the conditions in (2.1.3) with

$$\frac{1}{||\beta||} y_i(x_i^T \beta + \beta_0) \geq M, \ i = 1, ...N \ , \tag{2.1.4}$$

which leads to a redifinition of β_0.

This can be equivalently expressed as

$$y_i(x_i^T \beta + \beta_0) \geq M||\beta||, \ i = 1, ...N \ . \tag{2.1.5}$$

For any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, so we can arbitrarily set $||\beta|| = 1/M$ and get

$$\begin{aligned} \max_{\beta, \beta_0} \quad & M \\ \text{subject to} \quad & y_i(x_i^T \beta + \beta_0) \geq 1, \ i = 1, ...N \ , \end{aligned} \tag{2.1.6}$$

or equivalently,

$$\begin{aligned} \min_{\beta, \beta_0} \quad & ||\beta|| \\ \text{subject to} \quad & y_i(x_i^T \beta + \beta_0) \geq 1, \ i = 1, ...N \ . \end{aligned} \tag{2.1.7}$$

For convenience to compute derivatives, usually the following equivalent form of (2.1.7) is used:

$$\begin{aligned} \min_{\beta,\beta_0} \quad & \frac{1}{2}||\beta||^2 \\ \text{subject to} \quad & y_i(x_i^T\beta+\beta_0) \geq 1, i = 1,...N \ . \end{aligned} \tag{2.1.8}$$

In practice, the problem solved is the dual formulation derived using standard **Lagrangian techniques** [18]. To obtain it, first we get the **Lagrange primal problem**, which is

$$\begin{aligned} \min_{\beta,\beta_0} \quad & L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-1] \\ \text{subject to} \quad & \alpha_i \geq 0, \ i = 1,...N \ . \end{aligned} \tag{2.1.9}$$

Computing the derivatives in (2.1.9) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta - \sum_{i=1}^{N}\alpha_i y_i x_i \tag{2.1.10}$$

$$\frac{\partial L_P}{\partial \beta_0} = -\sum_{i=1}^{N}\alpha_i y_i \ , \tag{2.1.11}$$

and setting the derivatives (2.1.10) and (2.1.11) to zero we get:

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i \tag{2.1.12}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \ . \tag{2.1.13}$$

Plugging (2.1.12) and (2.1.13) into (2.1.9) we finally obtain the **dual problem**, with the called **Karush-Kuhn-Tucker, KKT, conditions** as constraints:

$$\begin{aligned} \max_{\alpha_i} \quad & L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \quad & y_i(x_i^T\beta+\beta_0) \geq 1, \ i = 1,...N \\ & \alpha_i \geq 0, \ i = 1,...N \\ & \beta = \sum_{i=1}^{N}\alpha_i y_i x_i \\ & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & \alpha_i[y_i(x_i^T\beta+\beta_0)-1] = 0 \ . \end{aligned} \tag{2.1.14}$$

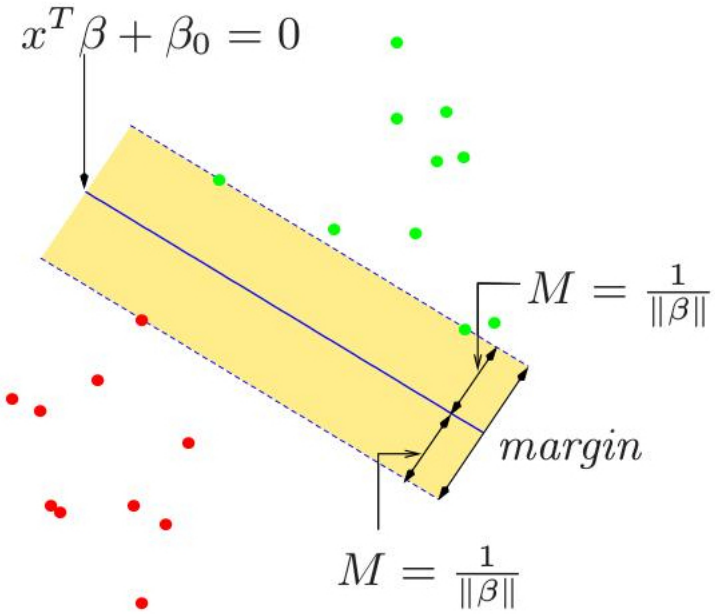


Figure 2.1.1: Support vector classifier for the separable case. The band in the figure is M units away from the hyperplane on either side, and hence $2M$ units wide. It is called the *margin*. Image from [5].

Thus, from (2.1.12) we see that the solution for β has the form

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \ , \tag{2.1.15}$$

where

$$\hat{\alpha}_i > 0 \Rightarrow y_i(x_i^T\beta + \beta_0) - 1 = 0 \ , \tag{2.1.16}$$

due to the KKT condition $\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0$.

These observations are called the **support vectors** and give name to the model, since $\hat{\beta}$ is represented only in terms of these points.

Finally, from the KKT condition $\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0$ we can see that any of these support vectors can be used to solve for β_0, which solution has the form

$$\hat{\beta}_0 = \frac{1 - y_i x_i^T \hat{\beta}}{y_i} \ . \tag{2.1.17}$$

Usually an average of the solutions for each support vector point is used for numerical stability.

2. Linear Non-Separable Case. Soft Margin Classification

In real-world problems, usually finding a hyperplane which separates perfectly the data is not possible. Figure 2.1.2 is an example of this type of dataset.

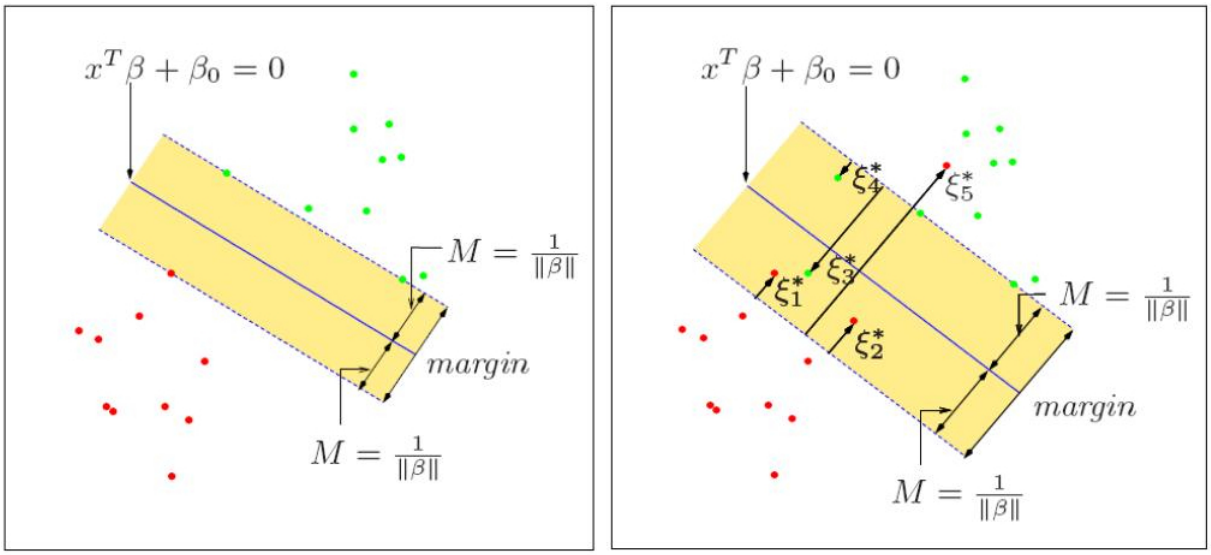


Figure 2.1.2: Linearly Separable (left) vs. Linearly Non-Separable (right). Image from [5].

Even if it is, it might not be desirable because the probability of the model overfitting the data, due to the noise or outliers present in the dataset, is rather high and normally a decision boundary that ignores some points of the dataset which do not represent the general behavior of the problem is preferred. An example of this problem, where a hard margin hyperplane suffers from overfitting due to one only noisy or outilier point, is shown in 2.1.3

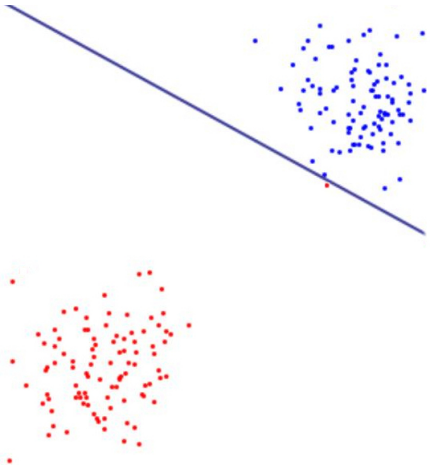


Figure 2.1.3: Hard Margin Classification overfitting.

To deal with the overlap, the idea is to still maximize the margin, M, but allowing some points of the dataset to be on the wrong side of the margin. Defining the slack variables $\xi = (\xi_1, \xi_2, ..., \xi_N)$, one natural way to modify the constraint in (2.1.3) will be

$$y_i(x_i^T\beta + \beta_0) \geq M - \xi_i,\ i = 1, ...N\ , \tag{2.1.18}$$

where the value ξ_i is the absolute value of the amount by which the prediction $f(x_i) = x_i^T\beta + \beta_0$ is on the wrong side of its margin.

Although this choice seems very natural, since it measures overlap in actual distance from the margin M, it results in a nonconvex optimization problem, so uniqueness of the solution cannot be assured. To find a convex optimization problem, where any local minimum of the unconstrained optimization problem is a global minimum and hence the solution is unique [1], the following modification is used:

$$y_i(x_i^T\beta + \beta_0) \geq M - M\xi_i = M(1 - \xi_i),\ i = 1, ...N\ , \tag{2.1.19}$$

where now ξ_i is the relative value with respect to M by which the prediction $f(x_i) = x_i^T\beta + \beta_0$ is on the wrong side of its margin.

With this formulation, misclassifications occur when $M\xi_i > M$, i.e. $\xi_i > 1$. Thus, bounding $\sum_{i=1}^N \xi_i$ at a value λ sets the upper bound of the total number of training misclassifications to be λ

Plugging (2.1.19) into (2.1.3) we get

$$\begin{aligned} \max_{\beta,\beta_0} \quad & M \\ \text{subject to} \quad & y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i),\ i = 1, ...N \\ & ||\beta|| = 1 \\ & \xi_i \geq 0,\ i = 1, ...N \\ & \sum_{i=1}^N \xi_i \leq \lambda\ . \end{aligned} \tag{2.1.20}$$

As we did in the separable case, we can drop the norm constraint on β of (2.1.20), define $M = \frac{1}{||\beta||}$, and write the equation in the equivalent form

$$\begin{aligned} \min_{\beta,\beta_0} \quad & \frac{1}{2}||\beta||^2 \\ \text{subject to} \quad & y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i,\ i = 1, ...N \\ & \xi_i \geq 0,\ i = 1, ...N \\ & \sum_{i=1}^N \xi_i \leq \lambda\ . \end{aligned} \tag{2.1.21}$$

[1] Recall that existence of the solution is guaranteed by the quadratic nature of the objective function.

As described in [5], computationally it is convenient to re-express (2.1.21) in the equivalent form

$$\begin{aligned} \min_{\beta,\beta_0} \quad & \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(x_i^T\beta+\beta_0) \geq 1-\xi_i,\ i=1,...N \\ & \xi_i \geq 0,\ i=1,...N\ , \end{aligned} \tag{2.1.22}$$

where the parameter C, often called the *cost*, replaces λ in (2.1.21). It is easy to see that the hard margin case corresponds to $C=\infty$ that leads to $\sum_{i=1}^{N}\xi=0$, i.e., not a single point is allowed to be on the wrong side of the margin M.

Equation in (2.1.22) is quadratic with linear inequality constraints, hence it is a convex optimization problem. As for the separable case, the problem solved in practice is the dual formulation derived using Lagrangian techniques. First, we get the Lagrange primal problem, that in this case has the form

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i} \quad & L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i \\ & -\sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i \\ \text{subject to} \quad & \alpha_i \geq 0,\ i=1,...N \\ & \mu_i \geq 0,\ i=1,...N\ . \end{aligned} \tag{2.1.23}$$

Computing the derivatives in (2.1.23) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta - \sum_{i=1}^{N}\alpha_i y_i x_i \tag{2.1.24}$$

$$\frac{\partial L_P}{\partial \beta_0} = -\sum_{i=1}^{N}\alpha_i y_i \tag{2.1.25}$$

$$\frac{\partial L_P}{\partial \xi_i} = C-\alpha_i-\mu_i,\ i=1,...N\ , \tag{2.1.26}$$

and setting the derivatives (2.1.24), (2.1.25) and (2.1.26) to zero we get:

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i \tag{2.1.27}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{2.1.28}$$

$$\alpha_i = C-\mu_i,\ i=1,...N\ . \tag{2.1.29}$$

Plugging (2.1.12) and (2.1.13) into (2.1.9) we finally obtain the **dual problem**, with the called **Karush-Kuhn-Tucker, KKT, conditions** as constraints:

$$\begin{aligned}
\max_{\alpha_i} \quad & L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to} \quad & y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i, \ i = 1, ...N \\
& \xi_i \geq 0 \\
& \alpha_i \geq 0, \ i = 1, ...N \\
& \mu_i \geq 0, \ i = 1, ...N \Rightarrow \ \alpha_i \leq C, \ i = 1, ...N \\
& \beta = \sum_{i=1}^{N} \alpha_i y_i x_i \\
& \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0 \\
& \mu_i \xi_i = 0, \ i = 1, ...N \Rightarrow \ (C - \alpha_i)\xi_i = 0 \ .
\end{aligned} \tag{2.1.30}$$

Thus, from (2.1.30) we see that, as in the case of the hard margin classificator, the solution for β has the form (2.1.15). In this case we have that support vectors are the points with $\alpha_i > 0$, characterized by

$$\hat{\alpha_i} > 0 \Rightarrow y_i(x_i^T \beta + \beta_0) - (1 - \xi_i) = 0 \ , \tag{2.1.31}$$

due to the KKT condition $\alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0$

Some of these support vectors will lie on the edge of the margin, i.e. $\hat{\xi}_i = 0$, and the remainder will be characterized by

$$\hat{\xi}_i > 0 \Rightarrow \hat{\alpha}_i = C \ , \tag{2.1.32}$$

due to the KKT condition $(C - \alpha_i)\xi_i = 0$.

Finally, from the KKT condition $\alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0$ we can see that the solution for β_0 can be obtained using any of the margin points and is given again by (2.1.17)

3. Non-Linear Non-Separable Case

The support vector classifier previously described finds linear boundaries in the input feature space. We can enlarge the feature space using basis expansions such as polynomials or splines, for example. Generally linear boundaries in the enlarged space achieve better class separation, and translate to nonlinear boundaries in the original feature space. Figure 2.1.4 is an example of this.

Once the basis functions $\{h_m(x)\}_{m=1}^{M}$ are selected, the procedure is the same as before. We fit the SVM using this time as input features $\{h(x_i)\}_{i=1}^{N} = \{(h_1(x_i), h_2(x_i), ..., h_M(x_i))\}_{i=1}^{N}$ instead of the original $\{x_i\}_{i=1}^{N}$, and produce the function

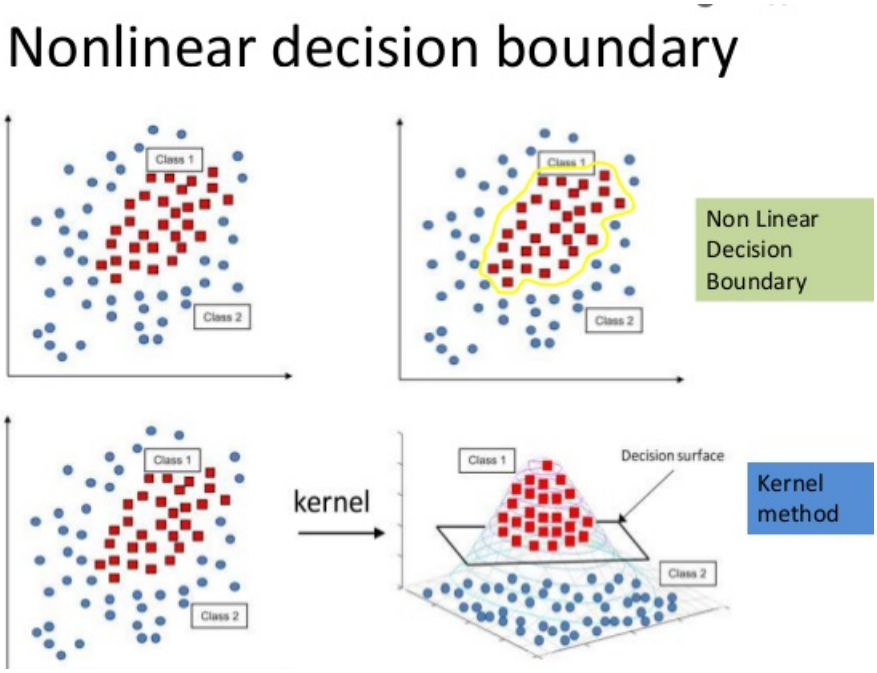


Figure 2.1.4: Non-linear SVM. Image from [19].

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0 \ , \tag{2.1.33}$$

where this time $\hat{f}(x)$ is a non-linear function.

Replacing $\{x_i\}_{i=1}^N$ for the new input features $\{h(x_i)\}_{i=1}^N$ in (2.1.30) we get the following dual function

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j h(x_i)^T h(x_j) \ . \tag{2.1.34}$$

Doing the same replacement in (2.1.27) we have

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i h(x_i) \ . \tag{2.1.35}$$

Furhermore, plugging (2.1.35) into (2.1.33) we obtain

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i h(x)^T h(x_i) + \hat{\beta}_0 \ . \tag{2.1.36}$$

Looking at (2.1.34) and (2.1.36) we can see that $h(x)$ is involved only through inner products, i.e. the expression $h(x)^T h(x_i)$. Thus, we do not need to specify explicitly the transformation $h(x)$, needing only to know the **kernel function**

$$K(x, x') = h(x)^T h(x') \ . \tag{2.1.37}$$

Thus, we can reformulate (2.1.34) and (2.1.36) as

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{2.1.38}$$

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i k(x, x_i) + \hat{\beta}_0 \ . \tag{2.1.39}$$

where the basis functions do not appear explicitly, but only through their inner product defined by the kernel k. This important property of support vector machines is often called **the kernel trick**.

Mercer's theorem gives the condition, known as **Mercer's condition**, that is sufficient for a function to be used as a kernel function:

Theorem 1. *Mercer's Theorem.*
If a scalar function $k(x_i, x_j)$ is positive semi-definite,i.e.

$$\int k(x_i, x_j) g(x_i) g(x_j) dx_i dx_j \geq 0 \ \forall \ g \in L_2$$

then there is a mapping function

$$\phi : \mathbb{R}^d \to F$$

with F a Hilbert space and it can always be decomposed as an inner product

$$K(x_i, x_j) = < \phi(x_i) \phi(x_j) >$$

where $\phi(x) \in F$

Thus, a function needs only to verify Mercer's condition, i.e. to be positive semi-definite, to be a valid kernel function.

One of the most popular functions used as kernel for SVM models is the Radial basis or **Gaussian Kernel**:

$$K(x, x') = e^{-\gamma ||x - x'||^2} \ . \tag{2.1.40}$$

This kernel trick allows us to make the dimension of the enlarged space very large, infinite in some cases, only defining a suitable kernel function. It might seem that the computational costs would become prohibitive and that, since perfect separation is often achievable in these enlarged spaces, overfitting would occur. Here is when the role of the cost parameter C is clearer. A large value of C will discourage any positive ξ_i, and lead to an overfit wiggly boundary in the original feature space, while a value of C too small will encourage a small value of $||\beta||^2$, which in turn causes $f(x)$ and hence the hyperplane to have less variance, tending to underfit the model. Figure 2.1.5 shows this phenomenon.

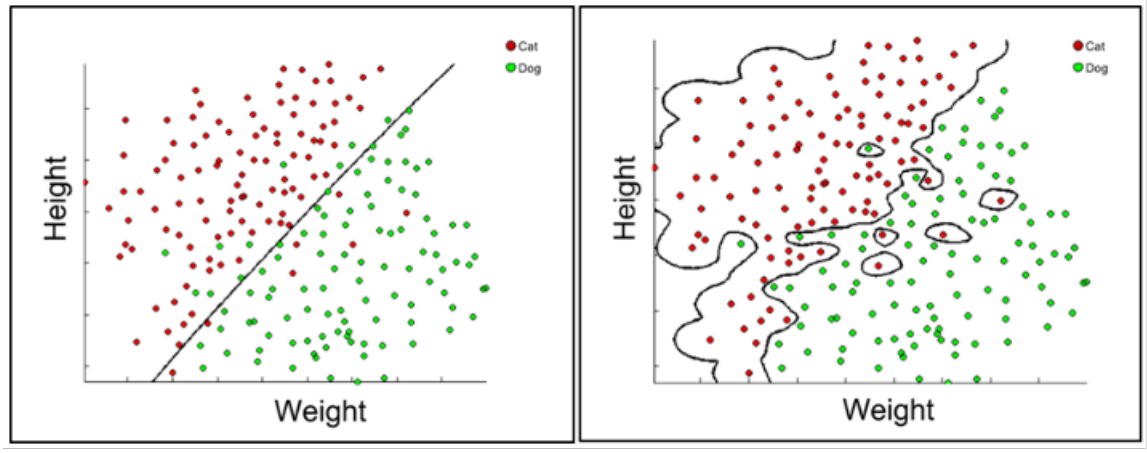


Figure 2.1.5: Low C values, (left) vs. High C values (right).

As mentioned in Section 1.1, a bias-variance tradeoff equilibrium between too low complexity, leading to too much bias and underfitting, and too high complexity, leading to too much variance and overfitting, must be reached. This is controlled through the parameter C, with large values of C producing high complexity models and small ones lower complexity models.

2.1.2 L1-ϵ-SVM. Regression

The support vector method can also be applied to the case of regression, maintaining all the main features that characterise the maximal margin algorithm. This method is called support vector regression, SVR. This section describe first the linear case and then this case is expanded to non-linear regressors using the analogous of the kernel trick described for classification.

1. Linear Case

In this case, the linear regression model is considered:

$$f(x) = x^T\beta + \beta_0 \ . \tag{2.1.41}$$

For SVR an objective function analogous to the one described in (1.1.3) is minimized, but this time with other loss function different to the quadratic error. The loss function used here is called the ϵ–**insensitive loss function, or ILF**

$$l_\epsilon(\delta) = \begin{cases} -\delta - \epsilon, & \delta < -\epsilon \\ 0, & \delta \in [-\epsilon, \epsilon] \\ \delta - \epsilon, & \delta > \epsilon \end{cases} \ . \tag{2.1.42}$$

The ILF function provides robustness against outliers. However, it is not only a robust cost function because of its linear behavior outside the interval $[-\epsilon, \epsilon]$, but also ignores the errors within a certain distance, ϵ, to the target value, y_i, assigning zero cost to errors

smaller than ϵ ,giving SVR a property known as **sparseness**. This can be seen in Figure 2.1.6.

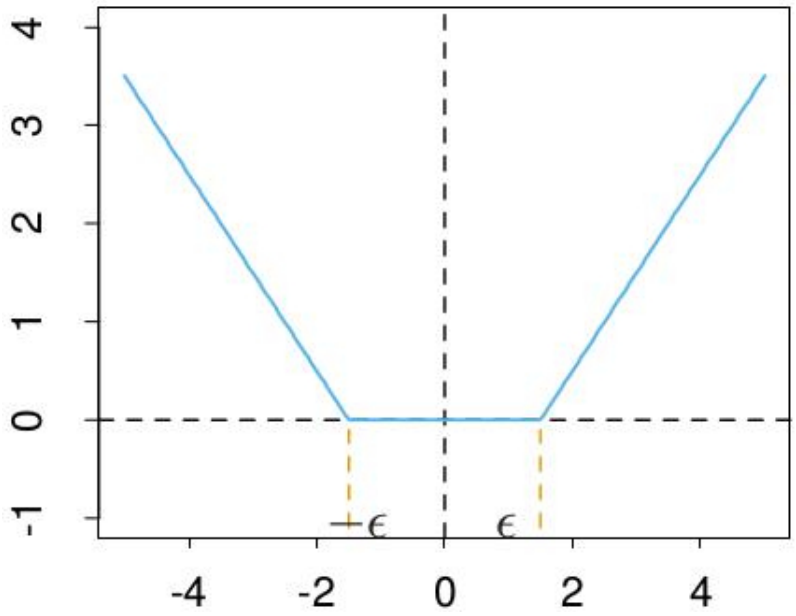


Figure 2.1.6: The ϵ–insensitive loss function. Image from [5].

The quadratic loss function is well justified under the assumption of Gaussian additive noise. However, the noise model underlying the choice of the ILF is not so clear. In [20], the use of the ILF is justified under the assumption that the noise is additive and Gaussian, where the variance and mean of the Gaussian are random variables.

This loss fuction is employed in combination with the ridge regression penalty to get the objective function and the optimization problem used in SVR:

$$\min_{\beta,\beta_0} \quad H(\beta,\beta_0) = \sum_{i=1}^{N} l_\epsilon(y_i - f(x)) + \frac{\lambda}{2}||\beta||^2 \ . \tag{2.1.43}$$

As stated in [5] and [17] this problem is equivalent to the following problem

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\hat{\xi}_i} \quad & \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}(\xi + \hat{\xi}_i) \\ \text{subject to} \quad & \xi_i, \hat{\xi}_i \geq 0, \ i = 1,...N \\ & f(x_i) - y_i \leq \epsilon + \xi_i, \ i = 1,...N \\ & y_i - f(x_i) \leq \epsilon + \hat{\xi}_i, \ i = 1,...N \ . \end{aligned} \tag{2.1.44}$$

where ξ_i quantifies the errors above the ϵ-band, and $\hat{\xi}_i$ the errors below the ϵ-band.

The Lagrange primal problem corresponding to 2.1.44 is

$$\begin{aligned}\min_{\beta,\beta_0,\xi_i,\hat{\xi}_i} \quad & L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) \\ & - \sum_{i=1}^{N}\alpha_i(y_i - f(x_i) + \epsilon + \xi_i) \\ & - \sum_{i=1}^{N}\alpha_i^*(f(x_i) - y_i + \epsilon + \hat{\xi}_i) \\ & - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\hat{\xi}_i \\ \text{subject to} \quad & \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\ & \mu_i, \mu_i^* \geq 0,\ i = 1, ...N\ .\end{aligned} \tag{2.1.45}$$

Computing the derivatives in (2.1.45) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta + \sum_{i=1}^{N}\alpha_i x_i - \sum_{i=1}^{N}\alpha_i^* x_i = \beta + \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)x_i \tag{2.1.46}$$

$$\frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) \tag{2.1.47}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i,\ i = 1, ...N \tag{2.1.48}$$

$$\frac{\partial L_P}{\partial \hat{\xi}_i} = C - \alpha_i^* - \mu_i^*,\ i = 1, ...N\ , \tag{2.1.49}$$

and setting the derivatives (2.1.46), (2.1.47), (2.1.48) and (2.1.49) to zero we get:

$$\beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)x_i \tag{2.1.50}$$

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \tag{2.1.51}$$

$$\alpha_i = C - \mu_i,\ i = 1, ...N \tag{2.1.52}$$

$$\alpha_i^* = C - \mu_i^*,\ i = 1, ...N\ . \tag{2.1.53}$$

Plugging (2.1.50), (2.1.51), (2.1.52) and (2.1.53) into (2.1.45) we get the dual problem, with the KKT conditions as constraints:

$$
\begin{aligned}
\max_{\alpha_i,\alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)x_i^T x_j - \epsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) \\
\text{subject to} \quad & f(x_i) - y_i \leq \epsilon + \xi_i,\ i = 1, ...N \\
& y_i - f(x_i) \leq \epsilon + \hat{\xi}_i,\ i = 1, ...N \\
& \xi_i, \hat{\xi}_i \geq 0 \\
& \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\
& \alpha_i, \alpha_i^* \leq C,\ i = 1, ...N \\
& \beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)x_i \\
& \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \\
& \alpha_i(y_i - f(x_i) + \epsilon + \xi_i) = 0 \\
& \alpha_i^*(f(x_i) - y_i + \epsilon + \hat{\xi}_i) = 0 \\
& (C - \alpha_i)\xi_i = 0 \\
& (C - \alpha_i^*)\hat{\xi}_i = 0\ .
\end{aligned}
\tag{2.1.54}
$$

The solution for β has the form

$$
\hat{\beta} = \sum_{i=1}^{N}(\hat{\alpha^*}_i - \hat{\alpha}_i)x_i\ . \tag{2.1.55}
$$

In this case we have that support vectors are the points where $(\alpha_i^* - \alpha_i) \neq 0$. This is equivalent to stating that the support vector points are the ones where $\alpha_i^* > 0$ or $\alpha_i > 0$, as both values cannot be different from zero for the same point, i.e. $\alpha_i\alpha_i^* = 0$, due to the conditions in (2.1.54).

Thus, the solution function can be shown to have the form

$$
\hat{f}(x) = \sum_{i=1}^{N}(\hat{\alpha^*}_i - \hat{\alpha}_i)x^T x_i + \hat{\beta}_0\ . \tag{2.1.56}
$$

Finally, the solution for β_0 can be obtained using any of the points where $\xi_i = 0$ or $\hat{\xi}_i = 0$ and the KKT conditions $\alpha_i(y_i - f(x_i) + \epsilon + \xi_i) = 0$ and $\alpha_i^*(f(x_i) - y_i + \epsilon + \hat{\xi}_i) = 0$ respectively, and has the form

$$
\hat{\beta}_0 = \begin{cases} y_i + \epsilon - x_i^T\hat{\beta} & \forall i\ with\ \alpha_i \in (0, C) \\ \\ y_i - \epsilon - x_i^T\hat{\beta} & \forall i\ with\ \alpha_i^* \in (0, C) \end{cases}\ . \tag{2.1.57}
$$

Figure 2.1.7 shows an example of a one dimensional linear regression function with an ϵ–insensitive band.

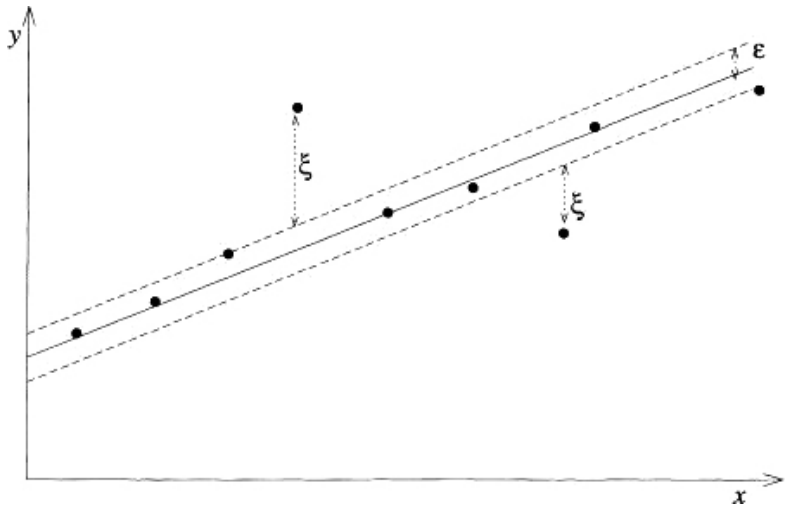


Figure 2.1.7: Linear SVR. Image from [17].

2. Non-linear Case. Kernel Trick

As in the classification case, we can enlarge the feature space to large dimensions, even infinite, through basis expansions. Replacing the original input features, $\{x_i\}_{i=1}^N$, with their corresponding features in the enlarged space, $\{h(x_i)\}_{i=1}^N$, in (2.1.54) we get the dual function

$$L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)h(x_i)^T h(x_j) - \epsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) \ . \tag{2.1.58}$$

Moreover, applying the same replacement approach to (2.1.56) we obtain

$$\hat{f}(x) = \sum_{i=1}^{N}(\hat{\alpha^*}_i - \hat{\alpha}_i)h(x)^T h(x_i) + \hat{\beta}_0 \ . \tag{2.1.59}$$

Once again, the values $h(x)$ only appear through their inner products, so using a function $k(x_i, x_j) = < h(x_i), h(x_j) >$ satisfying Mercer's condition we can get the following equivalent equations to (2.1.58) and (2.1.59)

$$L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)k(x_i, x_j) - \epsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) \tag{2.1.60}$$

$$\hat{f}(x) = \sum_{i=1}^{N}(\hat{\alpha^*}_i - \hat{\alpha}_i)k(x, x_i) + \hat{\beta}_0 \ , \tag{2.1.61}$$

with no need of explicit definition of the basis functions $\{h_m(x)\}_{m=1}^M$.

Figure 2.1.8 shows an example of a one dimensional non-linear regression function with an ϵ–insensitive band.

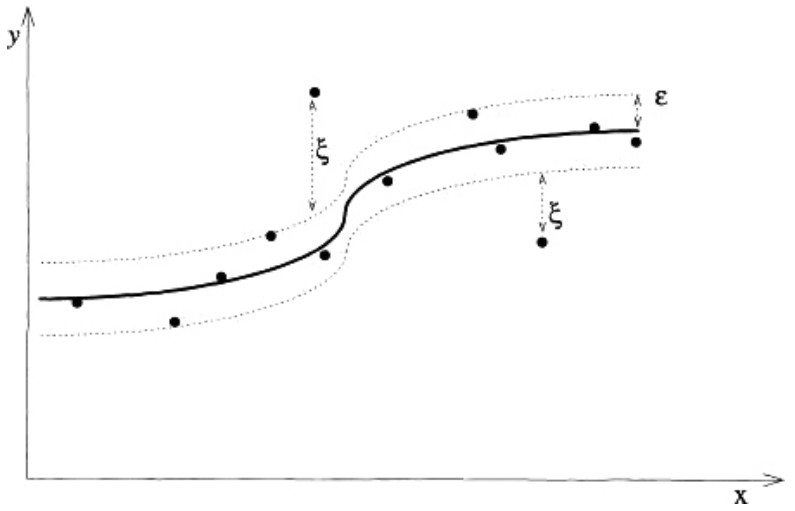


Figure 2.1.8: Non-linear SVR. Image from [17].

2.1.3 L2-SVR

Similarly to the definition in (2.1.42) for the ILF, the quadratic ϵ-insensitive loss function, or L2-ILF, is defined as follows

$$l2_\epsilon(\delta) = \begin{cases} (-\delta - \epsilon)^2, & \delta < -\epsilon \\ 0, & \delta \in [-\epsilon, \epsilon] \\ (\delta - \epsilon)^2, & \delta > \epsilon \end{cases} . \tag{2.1.62}$$

Figure 2.1.9 shows how this loss function looks. It has the same sparseness property as the standard ILF, or L1-ILF, since values in the interval $[-\epsilon, \epsilon]$ all are given zero value. Nonetheless, in contrast to the L1-ILF, this loss function is not robust to outliers, since a quadratic penalty is given to errors with absolute value greater than ϵ. This lack of robustness may suppose an important drawback for some problems, so appropriateness of the election of L2-SVR must be carefully considered.

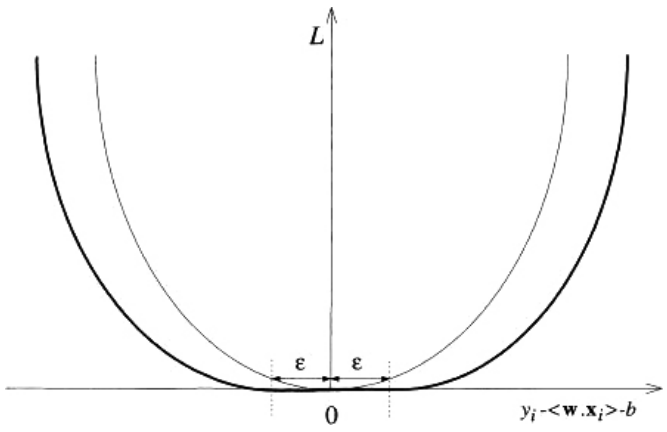


Figure 2.1.9: Quadratic ϵ-insensitive loss function. Image from [17].

Using this L2-ILF instead of the L1-ILF, we get the following Lagrange primal problem:

$$\begin{aligned}\min_{\beta,\beta_0,\xi_i,\hat{\xi}_i} \quad & L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}(\xi_i^2 + \hat{\xi}_i^2) \\ & - \sum_{i=1}^{N}\alpha_i(y_i - f(h(x_i)) + \epsilon + \xi_i) \\ & - \sum_{i=1}^{N}\alpha_i^*(f(h(x_i)) - y_i + \epsilon + \hat{\xi}_i) \\ & - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\hat{\xi}_i \\ \text{subject to} \quad & \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\ & \mu_i, \mu_i^* \geq 0,\ i = 1, ...N\ .\end{aligned} \tag{2.1.63}$$

Computing the derivatives in (2.1.63) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta + \sum_{i=1}^{N}\alpha_i h(x_i) - \sum_{i=1}^{N}\alpha_i^* h(x_i) = \beta + \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)h(x_i) \tag{2.1.64}$$

$$\frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) \tag{2.1.65}$$

$$\frac{\partial L_P}{\partial \xi_i} = 2C\xi_i - \alpha_i - \mu_i,\ i = 1, ...N \tag{2.1.66}$$

$$\frac{\partial L_P}{\partial \hat{\xi}_i} = 2C\hat{\xi}_i - \alpha_i^* - \mu_i^*,\ i = 1, ...N\ , \tag{2.1.67}$$

and setting the derivatives (2.1.64), (2.1.65), (2.1.66) and (2.1.67) to zero we get:

$$\beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)h(x_i) \tag{2.1.68}$$

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \tag{2.1.69}$$

$$\alpha_i = 2C\xi_i - \mu_i,\ i = 1, ...N \tag{2.1.70}$$

$$\alpha_i^* = 2C\hat{\xi}_i - \mu_i^*,\ i = 1, ...N\ . \tag{2.1.71}$$

Plugging (2.1.68), (2.1.69), (2.1.70) and (2.1.71) into (2.1.63) we get the dual problem, with the KKT conditions as constraints:

$$
\begin{aligned}
\max_{\alpha_i,\alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(k(x_i,x_j) + \frac{1}{C}\delta_{ij}) - \epsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) \\
\text{subject to} \quad & f(h(x_i)) - y_i \leq \epsilon + \xi_i,\ i = 1,...N \\
& y_i - f(h(x_i)) \leq \epsilon + \hat{\xi}_i,\ i = 1,...N \\
& \xi_i, \hat{\xi}_i \geq 0 \\
& \alpha_i, \alpha_i^* \geq 0,\ i = 1,...N \\
& \alpha_i = 2C\xi_i,\ i = 1,...N \\
& \alpha_i^* = 2C\hat{\xi}_i,\ i = 1,...N \\
& \beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)h(x_i) \\
& \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \\
& \alpha_i(y_i - f(h(x_i)) + \epsilon + \xi_i) = 0 \\
& \alpha_i^*(f(h(x_i)) - y_i + \epsilon + \hat{\xi}_i) = 0 \\
& (2C\xi_i - \alpha_i)\xi_i = 0 \\
& (2C\hat{\xi}_i - \alpha_i^*)\hat{\xi}_i = 0\ .
\end{aligned}
\tag{2.1.72}
$$

The support vectors are again the points with $(\alpha_i^* - \alpha_i) \neq 0$, or equivalently the ones where $\alpha_i^* > 0$ or $\alpha_i > 0$. The solutions for $\hat{\beta}$, $\hat{\beta}_0$ and $\hat{f}(x)$ have the same form described for L1-SVR in (2.1.55), (2.1.57) and (2.1.56) respectively.

2.1.4 ν-SVR

As it is difficult to select an appropiate value for the metaparameter ϵ in the L1-ϵ-SVR, in [21] and [22] a new parameter, ν, which lets one control the number of support vectors and training errors, is introduced, replacing ϵ in the dual formulation. Parameter ϵ is optimized automatically in this model.

This parameter ν is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors as proved in [21]. Thus, we can establish a priori a value for C, and then change ν to trade off the control model complexity.

The optimization problem is in this case

$$
\begin{aligned}
\min_{\beta,\beta_0,\xi_i,\hat{\xi}_i,\epsilon} \quad & \frac{1}{2}||\beta||^2 + C(\nu\epsilon + \frac{1}{N}\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i)) \\
\text{subject to} \quad & \xi_i, \hat{\xi}_i \geq 0,\ i = 1,...N \\
& \epsilon \geq 0,\ i = 1,...N \\
& f(x_i) - y_i \leq \epsilon + \xi_i,\ i = 1,...N \\
& y_i - f(x_i) \leq \epsilon + \hat{\xi}_i,\ i = 1,...N\ .
\end{aligned}
\tag{2.1.73}
$$

The corresponding primal problem for ν-SVR has the following form:

$$\begin{aligned}
\min_{\beta,\beta_0,\xi_i,\hat{\xi}_i,\epsilon} \quad & L_P = \frac{1}{2}||\beta||^2 + C(\nu\epsilon + \frac{1}{N}\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) \\
& - \sum_{i=1}^{N}\alpha_i(y_i - f(h(x_i)) + \epsilon + \xi_i) \\
& - \sum_{i=1}^{N}\alpha_i^*(f(h(x_i)) - y_i + \epsilon + \hat{\xi}_i) \\
& - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\hat{\xi}_i \\
& - \Gamma\epsilon \\
\text{subject to} \quad & \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\
& \mu_i, \mu_i^* \geq 0,\ i = 1, ...N \\
& \Gamma \geq 0\ .
\end{aligned} \tag{2.1.74}$$

Computing the derivatives in (2.1.74) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta + \sum_{i=1}^{N}\alpha_i h(x_i) - \sum_{i=1}^{N}\alpha_i^* h(x_i) = \beta + \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)h(x_i) \tag{2.1.75}$$

$$\frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) \tag{2.1.76}$$

$$\frac{\partial L_P}{\partial \xi_i} = \frac{C}{N} - \alpha_i - \mu_i,\ i = 1, ...N \tag{2.1.77}$$

$$\frac{\partial L_P}{\partial \hat{\xi}_i} = \frac{C}{N} - \alpha_i^* - \mu_i^*,\ i = 1, ...N \tag{2.1.78}$$

$$\frac{\partial L_P}{\partial \epsilon} = C\nu - \sum_{i=1}^{N}(\alpha_i + \alpha_i^*) - \Gamma\ , \tag{2.1.79}$$

and setting the derivatives (2.1.75), (2.1.76), (2.1.77), (2.1.78) and (2.1.79) to zero we get:

$$\beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)h(x_i) \tag{2.1.80}$$

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \tag{2.1.81}$$

$$\alpha_i = \frac{C}{N} - \mu_i,\ i = 1, ...N \tag{2.1.82}$$

$$\alpha_i^* = \frac{C}{N} - \mu_i^*,\ i = 1, ...N \tag{2.1.83}$$

$$\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) = C\nu - \Gamma\ . \tag{2.1.84}$$

Plugging (2.1.80), (2.1.81), (2.1.82), (2.1.83) and (2.1.84) into (2.1.74) we get the dual problem, with the KKT conditions as constraints:

$$
\begin{aligned}
\max_{\alpha_i,\alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)k(x_i, x_j) \\
\text{subject to} \quad & f(x_i) - y_i \leq \epsilon + \xi_i,\ i = 1, ...N \\
& y_i - f(x_i) \leq \epsilon + \hat{\xi}_i,\ i = 1, ...N \\
& \xi_i, \hat{\xi}_i \geq 0 \\
& \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\
& \alpha_i, \alpha_i^* \leq \frac{C}{N},\ i = 1, ...N \\
& \beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)h(x_i) \\
& \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \\
& \sum_{i=1}^{N}(\alpha_i + \alpha_i^*) \leq C\nu \\
& \alpha_i(y_i - f(x_i) + \epsilon + \xi_i) = 0 \\
& \alpha_i^*(f(x_i) - y_i + \epsilon + \hat{\xi}_i) = 0 \\
& (\frac{C}{N} - \alpha_i)\xi_i = 0 \\
& (\frac{C}{N} - \alpha_i^*)\hat{\xi}_i = 0\ .
\end{aligned}
\tag{2.1.85}
$$

In [23] authors show that the inequality constaint $\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) \leq C\nu$ can be replaced by an equality.

The support vectors are again the points with $(\alpha_i^* - \alpha_i) \neq 0$. The solutions for $\hat{\beta}$, $\hat{\beta}_0$ and $\hat{f}(x)$ are exactly equals to the ones described for L1-ϵ-SVR in (2.1.55), (2.1.57) and (2.1.56) respectively, but in the ν-SVR the tuning of ϵ dissapears and this parameter is replaced by ν, that might have a more clear interpretation, useful if parameter optimization is not carried out through and automatized process.

If we take the quadratic ϵ–insensitive loss instead of the ILF function, following an analogous process to the one described for the L2-ϵ-SVR we can obtain the L2-ν-SVR, which dual is identical to (2.1.85) except for three aspects:

- The quadratic term is changed to $-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(k(x_i, x_j) + \frac{N}{C}\delta_{ij})$.

- The Lagrangian coefficients α_i and α_i^* are now equal to $\frac{2C}{N}\xi_i$ and $\frac{2C}{N}\hat{\xi}_i$ respectively.

2.2 SMO

Maximizing the dual problems is a simpler convex quadratic programming problem than their corresponding primal problems, and can be solved with standard techniques. Usually the algorithm chosen for these problems is **Sequential Minimal Optimization**, or SMO [24].

2.2.1 SMO Description

Training a SVM model usually requires the solution of a very large quadratic programming, QP, optimization problem. SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop, as was the case in methods previous to the appearance of SMO.

In particular, for the standard SVM QP classification problem, the smallest possible optimization problem involves two Lagrange multipliers, because the Lagrange multipliers must obey the following linear equality constraint

$$\begin{cases} -\alpha_i + \alpha_j = \gamma & , \ y_i \neq y_j \\ \alpha_i + \alpha_j = \gamma & , \ y_i = y_j \end{cases} . \tag{2.2.1}$$

For the ϵ-SVR case, the smallest possible optimization problem implicates four Lagrange multipliers, satisfying this linear equality constraint:

$$(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = \gamma \ . \tag{2.2.2}$$

The Lagrange multipliers must fulfill all the constraints of the full problem, which means adding inequality constraints to the linear equality constraint already described. Tables 2.2.1 and 2.2.2 summarize all these constraints for the classification and regression case respectively. Table 2.2.2 has four cells, taking into account the fact that there are only four different possible pairs of nonzero variables since $\alpha_i \alpha_i^* = \alpha_j \alpha_j^* = 0$

Table 2.2.1: Boundary of feasible regions for classification.

$y_i = y_j$	$y_i \neq y_j$
$L_{clas} = max(0, -C + \gamma)$, $H_{clas} = min(C, \gamma)$	$L_{clas} = max(0, \gamma)$, $H_{clas} = min(C, C + \gamma)$

Table 2.2.2: Boundary of feasible regions for regression.

	$\alpha_j \neq 0$	$\alpha_j^* \neq 0$
$\alpha_i \neq 0$	$L_{reg} = max(0, -C + \gamma)$, $H_{reg} = min(C, \gamma)$	$L_{reg} = max(0, \gamma)$, $H_{reg} = min(C, C + \gamma)$
$\alpha_i^* \neq 0$	$L_{reg} = max(0, -\gamma)$, $H_{reg} = min(C, C - \gamma)$	$L_{reg} = max(0, -C - \gamma)$, $H_{reg} = min(C, -\gamma)$

Figure 2.2.1 shows the boundaries detailed before for the classification case. The inequality constraints cause the Lagrange multipliers to lie in a box. The linear equality constraint causes them to lie on a diagonal line. Therefore, combining them, one step of SMO must find an optimum of the objective function on a diagonal line segment.

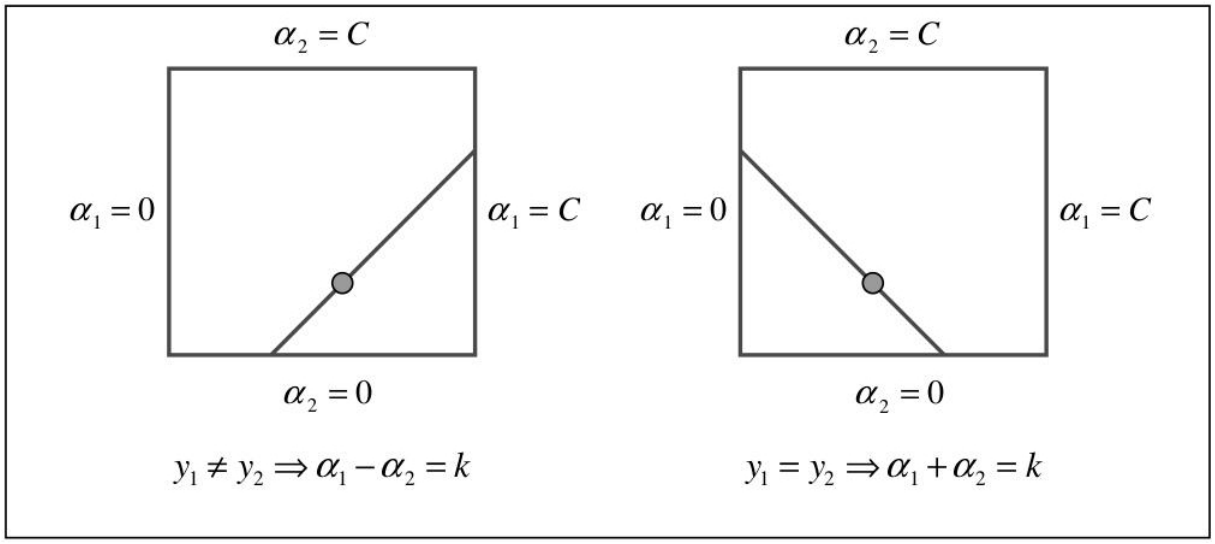


Figure 2.2.1: Boundary of feasible regions for classification. Image from [24].

At every step, SMO chooses two, for classification, or four, for regression, Lagrange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values. In each step, Lagrange multipliers must lie in the feasible regions described before.

The advantage of SMO with respect to previous methods lies in the fact that solving for the smallest possible optimization problem can be done analytically. Thus, numerical QP optimization is avoided entirely. Even though in total more optimization sub-problems are solved in the course of SMO, each sub-problem is so fast that the overall QP problem requires less computational time to be solved.

In addition, SMO requires no extra matrix storage at all. Thus, working with large scale SVM training datasets, typical in big data problems, is more feasible with this algorithm

and requires less powerful computing systems. Furthermore, because no matrix algorithms are used in SMO, it is less susceptible to numerical precision problems.

There are two main components to SMO: an analytic method for solving the smallest possible optimization problem, and a heuristic for choosing which subset of Lagrange multipliers to optimize.

2.2.2 Analytic Solution to the Smallest Possible Optimization Problem

A) Classification

We denote α_1 and α_2 the subset of Lagrange multipliers to optimize in a particular step of the algorithm. For convenience, all quantities that refer to the first multiplier will have a subscript 1, while all quantities that refer to the second multiplier will have a subscript 2. Without loss of generality, the algorithm first computes the new value of the second Lagrange multiplier, α_2^{new} and then uses it to obtain α_1^{new}.

The second derivative of the objective function along the diagonal line can be expressed as:

$$\eta = K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2) \ . \tag{2.2.3}$$

Under normal circumstances, η will be greater than zero. In this case, SMO computes the minimum along the direction of the constraint

$$\alpha_2^{new,noclipped} = \alpha_2 + \frac{y_2(\phi_1 - \phi_2)}{\eta} \ , \tag{2.2.4}$$

where $\phi_i = \hat{f}(x_i) - y_i$ is the error on the ith training example.

Now, the constrained minimum is found by clipping the unconstrained minimum to the end of the line segment:

$$\alpha_2^{new} = \begin{cases} H_{clas} & , \ \alpha_2^{new,noclipped} \geq H_{clas} \\ \alpha_2^{new,noclipped} & , L_{clas} < \alpha_2^{new,noclipped} < H_{clas} \\ L_{clas} & , \ \alpha_2^{new,noclipped} \leq L_{clas} \end{cases} \ . \tag{2.2.5}$$

Finally, α_1^{new} is computed from α_2^{new} through the following expression

$$\alpha_1^{new} = \alpha_1 + s(\alpha_2 - \alpha_2^{new}) \ , \tag{2.2.6}$$

where $s = y_1 y_2$.

Under unusual circumstances, η will not be positive. A negative η will occur only if the kernel K does not obey Mercer's condition. Thus, we will not consider this case, although a version of SMO with application to non-positive kernels is described in [25]. However, a zero η can occur even with a correct kernel satisfying Mercer's condition, if more than one training example has the same input vector x. In any event, SMO will work even when η is not positive, in which case the objective function L_D should be evaluated at each end of the line segment through the following computations:

$$\begin{aligned} g_1 &= y_1(\phi_1 + \beta_0) - \alpha_1 K(x_1, x_1) - s\alpha_2 K(x_1, x_2) \\ g_2 &= y_2(\phi_2 + \beta_0) - \alpha_2 K(x_2, x_2) - s\alpha_1 K(x_1, x_2) \\ L_1 &= \alpha_1 + s(\alpha_2 - L_{clas}) \\ H_1 &= \alpha_1 + s(\alpha_2 - H_{clas}) \end{aligned} \tag{2.2.7}$$

$$L_D^L = L_1 g_1 + L_{clas} g_2 + \frac{1}{2} L_1^2 K(x_1, x_1) + \frac{1}{2} L_{clas}^2 K(x_2, x_2) + s L_1 L_{clas} K(x_1, x_2)$$

$$L_D^H = H_1 g_1 + H_{clas} g_2 + \frac{1}{2} H_1^2 K(x_1, x_1) + \frac{1}{2} H_{clas}^2 K(x_2, x_2) + s H_1 H_{clas} K(x_1, x_2) \ .$$

SMO will move the Lagrange multipliers to the end point that has the lowest value of the objective function. If the objective function is the same at both ends then the joint minimization cannot make progress.

B) Regression

In this case, we have four Lagrange multipliers: α_1, α_1^*, α_2 and α_2^*, to optimize in a particular step of the algorithm. Again, for convenience all quantities that refer to the ith multiplier will have a subscript i. Without loss of generality, the algorithm first computes the new value of $\alpha_2^{(*)}$, and then uses them to obtain ${\alpha_1^{(*)}}^{new}$

These new values are obtained using the following steps, analogous to (2.2.4), (2.2.5) and (2.2.6)

$$\begin{cases} {\alpha_2}^{new,noclipped} = \alpha_2 + \frac{\phi_1 - \phi_2}{\eta} & , \ \alpha_1 \neq 0 \ , \ \alpha_2 \neq 0 \\ {\alpha_2^*}^{new,noclipped} = \alpha_2^* + \frac{\phi_1 - \phi_2 + 2\epsilon}{\eta} & , \ \alpha_1 \neq 0 \ , \ \alpha_2^* \neq 0 \\ {\alpha_2}^{new,noclipped} = \alpha_2^* + \frac{\phi_1 - \phi_2 - 2\epsilon}{\eta} & , \ \alpha_1^* \neq 0 \ , \ \alpha_2 \neq 0 \\ {\alpha_2^*}^{new,noclipped} = \alpha_2 + \frac{\phi_1 - \phi_2}{\eta} & , \ \alpha_1^* \neq 0 \ , \ \alpha_2^* \neq 0 \end{cases} \tag{2.2.8}$$

$${\alpha_2^{(*)}}^{new} = \begin{cases} H_{reg} & , \ {\alpha_2^{(*)}}^{new,noclipped} \geq H_{reg} \\ {\alpha_2^{(*)}}^{new,noclipped} & , L_{reg} < {\alpha_2^{(*)}}^{new,noclipped} < H_{reg} \\ L_{reg} & , \ {\alpha_2^{(*)}}^{new,noclipped} \leq L_{reg} \end{cases} \tag{2.2.9}$$

$$\begin{cases} \alpha_1^{new} = \alpha_1 + (-{\alpha_2}^{new} + \alpha_2) & , \ \alpha_1 \neq 0 \ , \ \alpha_2 \neq 0 \\ \alpha_1^{new} = \alpha_1 + (\alpha_2^{*new} - \alpha_2^*) & , \ \alpha_1 \neq 0 \ , \ \alpha_2^* \neq 0 \\ \alpha_1^{*new} = \alpha_1^* + (-{\alpha_2}^{new} + \alpha_2) & , \ \alpha_1^* \neq 0 \ , \ \alpha_2 \neq 0 \\ \alpha_1^{*new} = \alpha_1^* + (\alpha_2^{*new} - \alpha_2^*) & , \ \alpha_1^* \neq 0 \ , \ \alpha_2^* \neq 0 \end{cases} . \tag{2.2.10}$$

As in the classification case, zero η can occur even with a correct kernel satisfying Mercer's condition. In this case, the optimal value lies on the boundaries H_{reg} or L_{reg} . One can find out which one of the endpoints to take by looking at the gradient or simply by computing the value of the objective function at the endpoints as we described for the classification case.

2.2.3 Heuristic for Choosing Subset of Lagrange Multipliers

In order to speed convergence, SMO uses heuristics to choose which subset of Lagrange multipliers to jointly optimize. We will describe here one possible choice for these heuristics. Other choices for the heuristics to use can be found in [26].

A) Classification

In [24], heuristics for the classification case are proposed. There are two separate choice heuristics: one for the first Lagrange multiplier and one for the second. Without loss of generality, we set α_2 to be the first Lagrange multiplier and α_1 the second

The choice of the first heuristic provides the outer loop of the SMO algorithm. The outer loop first iterates over the entire training dataset, determining whether each example violates the KKT conditions and thus is eligible for optimization. After one pass through the entire training set, the outer loop iterates over all non-bound instances, i.e. those whose Lagrange multipliers are neither 0 nor C. Again, each example is checked against the KKT conditions and violating examples are marked as eligible for optimization. The outer loop makes repeated passes over the non-bound examples until all of them obey the KKT conditions within a certain threshold τ, with the value of τ typically chosen to be 10^{-3}. The outer loop then goes back and iterates over the entire training set again. The outer loop keeps alternating between single passes over the entire training set and multiple passes over the non-bound subset until the entire training set obeys the KKT conditions within τ.

Thus, the first choice heuristic concentrates the CPU time on the non-bound subset examples, that are most likely to violate the KKT conditions. As the SMO algorithm progresses, examples that are at the bounds are likely to stay there, while examples that are not at the bounds will move as other examples are optimized. The SMO algorithm will thus iterate over the non-bound subset until that subset is self-consistent, with all examples verifying the KKT conditions, and then SMO will scan the entire data set to search for any bound examples that have become KKT violated due to optimizing the non-bound subset.

Once a first Lagrange multiplier is chosen, SMO chooses the second Lagrange multiplier to maximize the size of the step, Δ, taken during joint optimization

$$\Delta = |\frac{y_2(\phi_1 - \phi_2)}{\eta}| \ . \tag{2.2.11}$$

Sometimes, in order to avoid the costly computation of K to obtain η, the following approximation, proposed in [24], of the step size is used instead

$$\hat{\Delta} = |\phi_1 - \phi_2| \ . \tag{2.2.12}$$

Under unusual circumstances, SMO cannot make positive progress using the second choice heuristic described above. For example, as described before, positive progress cannot be made if the first and second training examples share identical input vectors x, which causes the objective function to become semi-definite. When this occurs, SMO uses a hierarchy of second choice heuristics until it finds a pair of Lagrange multipliers that can be make positive progress. Positive progress can be determined by making a non-zero step size upon joint optimization of the subset of Lagrange multipliers. In this case, the Lagrange multiplier with next maximum step size can be chosen. Other option is to choose any Lagrange multiplier that can make positive progress, searching first in the non-bound examples and then in the bound ones, with both the iteration through the non-bound examples and the iteration through the entire training set starting at random locations, in order not to bias SMO towards the examples at the beginning of the training set. In extremely degenerate circumstances, none of the examples will make an adequate second example. When this happens, the first Lagrange multiplier chosen is skipped and SMO continues with another choice of α_2.

B) Regression

For the regression case, we will mimic the reasoning described for classification. The outer loop is exactly the same as the one for the classification problem. Again, once a first Lagrange multiplier is chosen, SMO chooses the second Lagrange multiplier to maximize the size of the step, Δ, taken during joint optimization

$$\Delta = \begin{cases} |\frac{\phi_1 - \phi_2}{\eta}| & , \ \alpha_1 \neq 0 \ , \ \alpha_2 \neq 0 \\ |\frac{\phi_1 - \phi_2 + 2\epsilon}{\eta}| & , \ \alpha_1 \neq 0 \ , \ \alpha_2^* \neq 0 \\ |\frac{\phi_1 - \phi_2 - 2\epsilon}{\eta}| & , \ \alpha_1^* \neq 0 \ , \ \alpha_2 \neq 0 \\ |\frac{\phi_1 - \phi_2}{\eta}| & , \ \alpha_1^* \neq 0 \ , \ \alpha_2^* \neq 0 \end{cases} , \tag{2.2.13}$$

or its approximated value

$$\hat{\Delta} = \begin{cases} |\phi_1 - \phi_2| & , \ \alpha_1 \neq 0 \ , \ \alpha_2 \neq 0 \\ |\phi_1 - \phi_2 + 2\epsilon| & , \ \alpha_1 \neq 0 \ , \ \alpha_2^* \neq 0 \\ |\phi_1 - \phi_2 - 2\epsilon| & , \ \alpha_1^* \neq 0 \ , \ \alpha_2 \neq 0 \\ |\phi_1 - \phi_2| & , \ \alpha_1^* \neq 0 \ , \ \alpha_2^* \neq 0 \end{cases} . \tag{2.2.14}$$

2.2.4 Parameters Update

After each joint optimization of a subset of Lagrange multipliers, the parameters β and β_0 must be updated. This computation depends on the nature of the problem:

A) Classification

The weight vector update is easy, due to the linearity of the SVM:

$$\beta^{new} = \beta + y_1(\alpha_1^{new} - \alpha_1)h(x_1) + y_2(\alpha_2^{new} - \alpha_2)h(x_2) \ . \tag{2.2.15}$$

For β_0, the following value is valid when α_1^{new} is not at the bounds

$$\beta_0^a = \beta_0 + \phi_1 + y_1(\alpha_1^{new} - \alpha_1)K(x_1, x_1) + y_2(\alpha_2^{new} - \alpha_2)K(x_1, x_2) \ , \tag{2.2.16}$$

and the following one is valid when α_2^{new} is not at the bounds

$$\beta_0^b = \beta_0 + \phi_2 + y_2(\alpha_2^{new} - \alpha_2)K(x_2, x_2) + y_1(\alpha_1^{new} - \alpha_1)K(x_1, x_2) \ . \tag{2.2.17}$$

When both β_0^a and β_0^b are valid, they are equal. When both new Lagrange multipliers are at bound and if L is not equal to H, then the whole interval between β_0^a and β_0^b consists of thresholds that are consistent with the KKT conditions and typically the value chosen is $\frac{\beta_0^a + \beta_0^b}{2}$.

In fact, the value we want to compute is $\hat{f}^{new}(x) = h(x)^T \hat{\beta}^{new} + \hat{\beta}_0^{\ new}$, where $\hat{\beta}_0^{\ new} = \beta_0^a$ or $\hat{\beta}_0^{\ new} = \beta_0^b$, and therefore, again, in the computation carried out the basis functions do not appear explicitly, but only through their inner product defined by the kernel k.

B) Regression

In this case, the computations required to update the model parameters depends on which is the pair of non-zero Lagrange multipliers:

$$\begin{cases} \beta^{new} = \beta - (\alpha_1^{new} - \alpha_1)h(x_1) - (\alpha_2^{new} - \alpha_2)h(x_2) & , \ \alpha_1 \neq 0 \ , \ \alpha_2 \neq 0 \\ \beta^{new} = \beta - (\alpha_1^{new} - \alpha_1)h(x_1) + (\alpha_2^{*new} - \alpha_2)h(x_2) & , \ \alpha_1 \neq 0 \ , \ \alpha_2^* \neq 0 \\ \beta^{new} = \beta + (\alpha_1^{*new} - \alpha_1)h(x_1) - (\alpha_2^{new} - \alpha_2)h(x_2) & , \ \alpha_1^* \neq 0 \ , \ \alpha_2 \neq 0 \\ \beta^{new} = \beta + (\alpha_1^{*new} - \alpha_1)h(x_1) + (\alpha_2^{*new} - \alpha_2)h(x_2) & , \ \alpha_1^* \neq 0 \ , \ \alpha_2^* \neq 0 \end{cases} . \quad (2.2.18)$$

To compute the new value of β_0, if at least one of the variables $\alpha_i^{(*)}$ and $\alpha_j^{(*)}$ is inside the boundaries, one can exploit (2.1.57). In the rare case that this does not happen, again there exists a whole interval of admissible thresholds and typically the halfway value is chosen.

As before, the value we want to compute is $\hat{f}^{new}(x) = h(x)^T \hat{\beta}^{new} + \hat{\beta}_0^{new}$, where the basis functions do not appear explicitly, but only through their inner product defined by the kernel k.

2.3 Bayesian SVR

In [27] and [28], Bayesian interpretations of SVR, or BSVR, are described. Using a Bayesian framework, these papers present a method to determine parameters in SVR by maximizing an evidence function, and at the same time derive a probability interval for prediction.

In this section the focus is on the computation of optimal parameters for BSVR models. Calculation of probability intervals using this Bayesian framework will be discussed in Section 3.1.

2.3.1 Bayesian Framework

In the Bayesian Approach the data of the regression problem, $D = \{(x_i, y_i) | x_i \in R^n, y_i \in R, i = 1...N\}$, is supposed to be collected from the model:

$$y_i = f(x_i) + \delta_i \ , \quad (2.3.1)$$

where δ_i are independent and identically distributed random noises and $f = [f(x_1), ..., f(x_N)]$ is the realization of a random field with a known prior probability. The posterior probability of f given the training data D can then be derived by Bayes' theorem:

$$P(f|D) = \frac{P(D|f)P(f)}{P(D)} \ , \quad (2.3.2)$$

where $P(f)$ is the prior probability of the random field and $P(D|f)$ is the conditional probability of the data D given the function values f.

In [28], the prior is assumed to be a zero mean Gaussian process with covariance matrix, K, with elements:

$$K_{x_i,x_j} = \kappa_0 \exp(-\frac{\kappa}{2}||x_i - x_j||^2) + \kappa_b \ , \tag{2.3.3}$$

where $\kappa > 0, \kappa_b > 0$ denotes the variance of the offset to the function $f(x)$ and $\kappa_0 > 0$ denotes the average power of $f(x)$.

Therefore, inserting (2.3.3) into the probability density function of a multivariate normal distribution, we can express the prior probability as:

$$P(f) = (2\pi)^{\frac{N}{2}} |K|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} f^T K^{-1} f\right) \ . \tag{2.3.4}$$

As the additive noise δ_i in (2.3.1) is supposed to be i.i.d with probability distribution $P(\delta_i)$, the probability $P(D|f)$, or likelihood, can be evaluated by:

$$P(D|f) = \prod_{i=1}^{N} P(y_i - f(x_i)) = \prod_{i=1}^{N} P(\delta_i) \ . \tag{2.3.5}$$

$P(\delta_i)$ is often assumed to be of the exponential form such that:

$$P(\delta_i) \propto \exp\left(-C \cdot l(\delta_i)\right) \ , \tag{2.3.6}$$

where $C > 0$ and l is a given loss function.

Inserting (2.3.6) into (2.3.5) we obtain:

$$P(D|f) = \prod_{i=1}^{N} P(\delta_i) \propto \exp\left(-C \cdot \sum_{i=1}^{N} l(\delta_i)\right) = \exp\left(-C \cdot \sum_{i=1}^{N} l(y_i - f(x_i))\right) \ . \tag{2.3.7}$$

As loss function, [28] proposes the use of a novel loss function called the soft insensitive loss function, **SILF**:

$$l_{\epsilon,\beta}(\delta) = \begin{cases} -\delta - \epsilon, & \delta \in \Delta_{C^*} \\ \frac{(\delta + (1-\beta)\epsilon)^2}{4\beta\epsilon}, & \delta \in \Delta_{M^*} \\ 0, & \delta \in \Delta_0 \\ \frac{(\delta - (1-\beta)\epsilon)^2}{4\beta\epsilon}, & \delta \in \Delta_M \\ \delta - \epsilon, & \delta \in \Delta_C \end{cases} \ , \tag{2.3.8}$$

where $0 < \beta \leq 1$, $\epsilon > 0$, $\Delta_{C^*} = (-\infty, -(1+\beta)\epsilon)$, $\Delta_{M^*} = [-(1+\beta)\epsilon, -(1-\beta)\epsilon]$, $\Delta_0 = (-(1-\beta)\epsilon, (1-\beta)\epsilon)$, $\Delta_M = [(1-\beta)\epsilon, (1+\beta)\epsilon]$ and $\Delta_C = ((1+\beta)\epsilon, +\infty)$

The purpose of using SILF as loss function is to combine in a single loss function two properties:

- The **sparseness** of the ϵ-insensitive loss function, which means that training samples with small noise that fall in the flat zero region are not involved in the representation of regression functions and therefore the computational cost is reduced.

- The **smoothness** similar to that of the quadratic and Huber's loss functions, that allows appropriate approximations to be used in the Bayesian approach.

2.3.2 Hyperparameter Selection

The parameters in the prior and the likelihood, $\theta = (\kappa, \kappa_0, \kappa_b, C, \epsilon)$, are called hyperparameters. The optimal values of hyperparameters θ can be inferred by maximizing the posterior probability $P(\theta|D)$:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \ . \tag{2.3.9}$$

In the approach proposed in [28], a flat distribution is assumed for $P(\theta)$, i.e., $P(\theta)$ is supposed to be greatly insensitive to the values of the hyperparameters θ. Thus, $P(D|\theta)$ can be used to assign a preference to alternative values of θ. Therefore, hyperparameters θ can be optimized by maximizing the evidence function

$$P(D|\theta) = \int P(D|f,\theta)P(f|\theta)df \ . \tag{2.3.10}$$

2.3.3 Bayesian Support Vector Regression

Now, maximizing the posterior probability in (2.3.2) is equivalent to maximize the numerator, so we have the following optimization problem:

$$\max_f \quad P(D|f)P(f) \ . \tag{2.3.11}$$

Plugging (2.3.7), (2.3.4) and (2.3.8) into (2.3.11) we get:

$$\max_f \quad \exp\left(-C \cdot \sum_{i=1}^{N} l_{\epsilon,\beta}(y_i - f(x_i))\right)(2\pi)^{\frac{N}{2}}|K|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}f^T K^{-1} f\right) = \tag{2.3.12}$$

$$\max_f \quad \exp\left(-C \cdot \sum_{i=1}^{N} l_{\epsilon,\beta}(y_i - f(x_i)) - \frac{1}{2}f^T K^{-1} f\right) = \tag{2.3.13}$$

$$\max_f \quad \exp\left(-S(f)\right) = \tag{2.3.14}$$

$$\min_f \quad S(f) \ , \tag{2.3.15}$$

where $S(f) = C \cdot \sum_{i=1}^{N} l_{\epsilon,\beta}(y_i - f(x_i)) + \frac{1}{2}f^T K^{-1} f$.

As usual, by introducing two slack variables ξ_i and ξ_i^*, (2.3.15) can be restated as the following equivalent optimization problem, known as the primal problem:

$$\begin{aligned} \min_{f,\xi_i,\xi_i^*} \quad & S(f,\xi_i,\xi_i^*) = C\sum_{i=1}^{N}(\psi(\xi_i)+\psi(\xi_i^*)) + \frac{1}{2}f^T K^{-1} f \\ \text{subject to} \quad & y_i - f(x_i) \leq (1-\beta)\epsilon + \xi_i \\ & f(x_i) - y_i \leq (1-\beta)\epsilon + \xi_i^* \\ & \xi_i \geq 0, i = 1,...N \\ & \xi_i \geq 0, i = 1,...N \ , \end{aligned} \tag{2.3.16}$$

where

$$\psi(\pi) = \begin{cases} \frac{\pi^2}{4\beta\epsilon}, & \pi \in [0, 2\beta\epsilon) \\ \\ \pi - \beta\epsilon, & \pi \in [2\beta\epsilon, +\infty) \end{cases} . \tag{2.3.17}$$

As in classical SVR, in practice the problem solved is the dual formulation, obtained using Lagrangian techniques:

$$\begin{aligned} \min_{\alpha,\alpha^*} \quad & S(\alpha,\alpha^*) = \frac{\beta\epsilon}{C}\sum_{i=1}^{N}(\alpha_i^2 + \alpha_i^{*2}) + \sum_{i=1}^{N}(\alpha_i + \alpha_i^*)(1-\beta)\epsilon \\ & + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i,x_j) - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)y_i \\ \text{subject to} \quad & 0 \leq \alpha_i <= C, i = 1,...N \\ & 0 \leq \alpha_i^* <= C, i = 1,...N \ . \end{aligned} \tag{2.3.18}$$

2.4 General Noise SVR. Proposed Approach

As explained before, the use of the ϵ–insensitive loss function in the classical SVR implies the assumption of a particular error distribution in the data [20]. However, it has been observed that the noise in some real-world applications, such as wind power forecasting, satisfies other distributions, including the Beta distribution [29],[30], the Weibull distribution [31] or the Laplacian distribution [32], to name a few.

In this section, we look to build a general noise formulation for SVR, where a particular distribution, p, for the noise is assumed, the optimal loss function for that distribution is computed and then plugged into the model to obtain a $p - SVR$ formulation for that distribution assumption.

2.4.1 General Noise Optimization Problem

In 2002, an ϵ–SVR for a general noise model was proposed in [33]. This general SVR can be used with any particular loss function $c(x_i, y_i, f(x))$, such as the gaussian loss function $c(x_i, y_i, f(x)) = (f(x_i) - y_i)^2$. Its optimization problem is described as:

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\hat{\xi}_i} \quad & \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}(c_i(\xi_i) + c_i(\hat{\xi}_i)) \\ \text{subject to} \quad & \xi_i, \hat{\xi}_i \geq 0,\ i = 1, ...N \\ & f(x_i) - y_i \leq \epsilon_i + \xi_i,\ i = 1, ...N \\ & y_i - f(x_i) \leq \epsilon_i^* + \hat{\xi}_i,\ i = 1, ...N\ , \end{aligned} \tag{2.4.1}$$

where ϵ_i, ϵ_i^* are chosen such that $c(xi, y_i, y_i + \xi) = 0\ ,\ \forall\ \xi \in [-\epsilon_i^*, \epsilon_i]$, $c_i(\xi_i) = c(xi, y_i, y_i + \epsilon_i + \xi)$ and $c_i(\hat{\xi}_i) = c(xi, y_i, y_i - \epsilon_i^* - \hat{\xi})$.

Note that c_i is the ϵ-clipped version of the original loss function.

2.4.2 Optimal Loss Function

Now, we want to obtain the optimal cost function in a maximum likelihood sense for a particular choice of error distribution in the data. Following [30], the general approach is to minimize

$$H[f] = \sum_{i=1}^{N} c(\xi_i) + \lambda\Phi[f]\ , \tag{2.4.2}$$

where λ is a positive number and $\Phi[f]$ is a smoothness functional.

We assume the noise is additive

$$y_i = f(x_i) + \xi_i,\ i = 1, ...N\ , \tag{2.4.3}$$

where ξ_i are independent and identically distributed, i.i.d., random variables.

A probabilistic approach is now taken, and the function f is regarded as the realization of a random field with a known prior probability distribution, $P[f]$. The goal is to maximize the posterior probability of f given the data D, $P[f|D]$. Using the Bayes Theorem we arrived to the following form for this probability

$$P[f|D] = \frac{P[D|f]P[f]}{P[D]} \propto P[D|f]P[f]\ , \tag{2.4.4}$$

where $P[D|f]$ is the conditional probability of the data D given the function f.

$P[D|f]$ is essentially a model of the noise, and if this noise is assumed to be additive, as in (2.4.3), and i.i.d. with probability distribution $P(\xi_i)$, this conditional probability can be written as

$$P[D|f] = \prod_{i=1}^{N} P(\xi_i)\ . \tag{2.4.5}$$

As explained in [30] and detailed in Section 2.3, the prior is often written as

$$P[f] \propto e^{-\lambda\Phi[f]}\ . \tag{2.4.6}$$

Now, replacing (2.4.6) and (2.4.5) into (2.4.4) we have

$$P[f|D] \propto e^{-\lambda\Phi[f]} \prod_{i=1}^{N} P(\xi_i) \ . \tag{2.4.7}$$

We want to maximize $P[f|D]$ or equivalently to minimize $-\log(P[f|D])$

$$\begin{aligned} -\log(P[f|D]) &\propto -\log(e^{-\lambda\Phi[f]} \prod_{i=1}^{N} P(\xi_i)) = \\ &= -\log(e^{-\lambda\Phi[f]}) - \log(\prod_{i=1}^{N} P(\xi_i)) = \\ &= \lambda\Phi[f] - \sum_{i=1}^{N} \log P(\xi_i) \ . \end{aligned} \tag{2.4.8}$$

Combining equations (2.4.2) and (2.4.8) we arrive to the conclusion that the optimal loss function in a maximum likelihood sense for a given error distribution, $P(\xi_i)$, is

$$c(\xi_i) = -\log P(\xi_i) = -\log P(y_i - f(x_i)) \ . \tag{2.4.9}$$

Thus, we can obtain now the optimal loss function for a given choice of noise distribution. However, the cost function resulting from this reasoning might be nonconvex. In this case, one may have to find a convex proxy in order to deal with the optimization problem or use a non-convex optimization method, such as the one proposed in [34] for SVMs.

In this work, we focus on the following distributions:

1. Zero-mean Laplace

The error distribution is assumed to be:

$$P(\xi_i) = \frac{1}{2\sigma} e^{-\frac{|\xi_i|}{\sigma}} \ , \tag{2.4.10}$$

where $\sigma > 0$.

Replacing (2.4.10) into (2.4.9) we get

$$\begin{aligned} c(\xi_i) &= -\log(\frac{1}{2\sigma} e^{-\frac{|\xi_i|}{\sigma}}) = \\ &= -\log(\frac{1}{2\sigma}) - \log(e^{-\frac{|\xi_i|}{\sigma}}) = \\ &= K + \frac{|\xi_i|}{\sigma} \ . \end{aligned} \tag{2.4.11}$$

As K is independent of ξ_i and therefore a constant for any error value, we ignore it from now on and work with the following expression

$$c(\xi_i) = \frac{|\xi_i|}{\sigma} \ . \tag{2.4.12}$$

2. General Laplace

The error distribution is assumed to be:

$$P(\xi_i) = \frac{1}{2\sigma} e^{-\frac{|\xi_i - \mu|}{\sigma}} \ , \tag{2.4.13}$$

where $\sigma > 0$ and $\mu \in (-\infty, \infty)$.

Replacing (2.4.13) into (2.4.9) we get

$$\begin{aligned} c(\xi_i) &= -\log\left(\frac{1}{2\sigma} e^{-\frac{|\xi_i - \mu|}{\sigma}}\right) = \\ &= -\log\left(\frac{1}{2\sigma}\right) - \log\left(e^{-\frac{|\xi_i - \mu|}{\sigma}}\right) = \\ &= K + \frac{|\xi_i - \mu|}{\sigma} \ . \end{aligned} \tag{2.4.14}$$

K is again independent of ξ_i so we ignore it and work with the following expression

$$c(\xi_i) = \frac{|\xi_i - \mu|}{\sigma} \ . \tag{2.4.15}$$

3. Zero-mean Gaussian

The error distribution is assumed to be:

$$P(\xi_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\xi_i^2}{2\sigma^2}} \ , \tag{2.4.16}$$

where $\sigma^2 > 0$.

Replacing (2.4.16) into (2.4.9) we get

$$\begin{aligned} c(\xi_i) &= -\log\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\xi_i^2}{2\sigma^2}}\right) = \\ &= -\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \log\left(e^{-\frac{\xi_i^2}{2\sigma^2}}\right) = \\ &= K + \frac{\xi_i^2}{2\sigma^2} \ . \end{aligned} \tag{2.4.17}$$

K is again independent of ξ_i so we ignore it and work with the following expression

$$c(\xi_i) = \frac{\xi_i^2}{2\sigma^2} \ . \tag{2.4.18}$$

4. General Gaussian

The error distribution is assumed to be:

$$P(\xi_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\xi_i-\mu)^2}{2\sigma^2}} \ , \tag{2.4.19}$$

where $\sigma^2 > 0$ and $\mu \in (-\infty, \infty)$.

Replacing (2.4.19) into (2.4.9) we get

$$\begin{aligned} c(\xi_i) &= -\log\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\xi_i-\mu)^2}{2\sigma^2}}\right) = \\ &= -\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \log\left(e^{-\frac{(\xi_i-\mu)^2}{2\sigma^2}}\right) = \\ &= K + \frac{(\xi_i-\mu)^2}{2\sigma^2} \ . \end{aligned} \tag{2.4.20}$$

K is again independent of ξ_i so we ignore it and work with the following expression

$$c(\xi_i) = \frac{(\xi_i-\mu)^2}{2\sigma^2} \ . \tag{2.4.21}$$

5. Beta

The error distribution is assumed to be:

$$P(\xi_i) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \xi_i^{\alpha-1}(1-\xi_i)^{\beta-1} \ , \tag{2.4.22}$$

where $\alpha, \beta > 0$ and $\Gamma(x)$ the gamma function.

Replacing (2.4.22) into (2.4.9) we get

$$\begin{aligned} c(\xi_i) &= -\log\Gamma(\alpha+\beta) + \log\Gamma(\alpha)\Gamma(\beta) - \log\xi_i^{\alpha-1} - \log\left(1-\xi_i\right)^{\beta-1} = \\ &= \log\Gamma(\alpha) + \log\Gamma(\beta) - \log\Gamma(\alpha+\beta) - (\alpha-1)\log\xi_i - (\beta-1)\log\left(1-\xi_i\right) = \\ &= K + (1-\alpha)\log\xi_i + (1-\beta)\log\left(1-\xi_i\right) \ . \end{aligned} \tag{2.4.23}$$

K is again independent of ξ_i so we ignore it and work with the following expression

$$c(\xi_i) = (1-\alpha)\log\xi_i + (1-\beta)\log\left(1-\xi_i\right) \ . \tag{2.4.24}$$

6. Weibull

The error distribution is assumed to be:

$$P(\xi_i) = \begin{cases} \frac{\kappa}{\lambda}\left(\frac{\xi_i}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{\xi_i}{\lambda}\right)^{\kappa}} & , \xi_i > 0 \\ 0 & , \xi_i \leq 0 \end{cases} \ , \tag{2.4.25}$$

where $\lambda, \kappa > 0$

Replacing the equation for $\xi_i > 0$ in (2.4.25) into (2.4.9) we get

$$\begin{aligned} c(\xi_i) &= -\log\frac{\kappa}{\lambda} - \log(\frac{\xi_i}{\lambda})^{\kappa-1} - \log e^{-(\frac{\xi_i}{\lambda})^{\kappa}} = \\ &= -\log\kappa + \log\lambda - (\kappa-1)\log\xi_i + (\kappa-1)\log\lambda + (\frac{\xi_i}{\lambda})^{\kappa} = \\ &= K + (1-\kappa)\log\xi_i + (\frac{\xi_i}{\lambda})^{\kappa} \ . \end{aligned} \tag{2.4.26}$$

K is again independent of ξ_i so we ignore it and work with the following expression

$$c(\xi_i) = \begin{cases} (1-\kappa)\log\xi_i + (\frac{\xi_i}{\lambda})^{\kappa} & ,\xi_i > 0 \\ 0 & ,\xi_i \leq 0 \end{cases} \ . \tag{2.4.27}$$

A summary of the loss functions we have detailed until now can be seen in Table 2.4.1

Table 2.4.1: Loss functions corresponding to several error distributions.

Error Distribution	Loss Function
ILF	$c(\xi_i) = \begin{cases} -\delta - \epsilon, & \delta < -\epsilon \\ 0, & \delta \in [-\epsilon, \epsilon] \\ \delta - \epsilon, & \delta > \epsilon \ . \end{cases}$
SILF	$c(\xi_i) = \begin{cases} -\delta - \epsilon, & \delta \in \Delta_{C^*} \\ \frac{(\delta+(1-\beta)\epsilon)^2}{4\beta\epsilon}, & \delta \in \Delta_{M^*} \\ 0, & \delta \in \Delta_0 \\ \frac{(\delta-(1-\beta)\epsilon)^2}{4\beta\epsilon}, & \delta \in \Delta_M \\ \delta - \epsilon, & \delta \in \Delta_C \ , \end{cases}$
Zero-mean Laplace	$c(\xi_i) = \frac{\lvert\xi_i\rvert}{\sigma}$
Laplace	$c(\xi_i) = \frac{\lvert\xi_i-\mu\rvert}{\sigma}$
Zero-mean Gaussian	$c(\xi_i) = \frac{\xi_i^2}{2\sigma^2}$
Gaussian	$c(\xi_i) = \frac{(\xi_i-\mu)^2}{2\sigma^2}$
Beta	$c(\xi_i) = (1-\alpha)\log\xi_i + (1-\beta)\log(1-\xi_i)$
Weibull	$c(\xi_i) = \begin{cases} (1-\kappa)\log\xi_i + (\frac{\xi_i}{\lambda})^{\kappa} & ,\xi_i > 0 \\ 0 & ,\xi_i \leq 0 \end{cases}$

2.4.3 General Noise Dual Problem

Now, we are ready to use (2.4.1) in combination with the loss functions in Table 2.4.1 to obtain the corresponding dual formulations for each error distribution assumption.

For some of these loss functions we will use (2.4.1) and make all the computations necessary to obtain the dual formulation. For conciseness, for the Beta and Weibull distributions we will use the following dual general problem derived from (2.4.1) .

$$
\begin{aligned}
\max_{\alpha_i,\alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i, x_j) \\
& - \sum_{i=1}^{N}\epsilon_i\alpha_i - \sum_{i=1}^{N}\epsilon_i^*\alpha_i^* \\
& + C\sum_{i=1}^{N}(T_i(\xi_i) + T_i^*(\hat{\xi}_i)) \\
\text{subject to} \quad & f(x_i) - y_i \leq \epsilon_i + \xi_i,\ i = 1, ...N \\
& y_i - f(x_i) \leq \epsilon_i^* + \hat{\xi}_i,\ i = 1, ...N \\
& \xi_i, \hat{\xi}_i \geq 0 \\
& \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\
& \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \\
& \alpha_i, \alpha_i^* \leq C\frac{\partial c(\xi)}{\partial \xi},\ i = 1, ...N \\
& \alpha_i(y_i - f(x_i) + \epsilon_i + \xi_i) = 0 \\
& \alpha_i^*(f(x_i) - y_i + \epsilon_i^* + \hat{\xi}_i) = 0 \\
& (C\frac{\partial c(\xi)}{\partial \xi} - \alpha_i)\xi_i = 0 \\
& (C\frac{\partial c(\xi)}{\partial \xi} - \alpha_i^*)\hat{\xi}_i = 0\ ,
\end{aligned}
\tag{2.4.28}
$$

where

$$
T_i(\xi_i) = c(\xi_i) - \xi_i\frac{\partial c(\xi)}{\partial \xi} \tag{2.4.29}
$$

$$
T_i^*(\xi_i) = c(\hat{\xi}_i) - \hat{\xi}_i\frac{\partial c(\hat{\xi})}{\partial \xi}\ . \tag{2.4.30}
$$

The proof for this general dual formulation can be found in [35].

1. SILF

The loss function here is

$$c(x_i, y_i, f(x_i)) = \begin{cases} -(f(x_i) - y_i) - \epsilon, & (f(x_i) - y_i) \in \Delta_{C^*} \\ \frac{((f(x_i)-y_i)+(1-\rho)\epsilon)^2}{4\rho\epsilon}, & (f(x_i) - y_i) \in \Delta_{M^*} \\ 0, & (f(x_i) - y_i) \in \Delta_0 \\ \frac{((f(x_i)-y_i)-(1-\rho)\epsilon)^2}{4\rho\epsilon}, & (f(x_i) - y_i) \in \Delta_M \\ (f(x_i) - y_i) - \epsilon, & (f(x_i) - y_i) \in \Delta_C \ , \end{cases} \quad , \tag{2.4.31}$$

where $0 < \rho \leq 1$, $\epsilon > 0$, $\Delta_{C^*} = (-\infty, -(1+\rho)\epsilon)$, $\Delta_{M^*} = [-(1+\rho)\epsilon, -(1-\rho)\epsilon]$, $\Delta_0 = (-(1-\rho)\epsilon, (1-\rho)\epsilon)$, $\Delta_M = [(1-\rho)\epsilon, (1+\rho)\epsilon]$ and $\Delta_C = ((1+\rho)\epsilon, +\infty)$

Using the conditions detailed in Section 2.4.1 we get

$$c(xi, y_i, y_i + \xi) = 0 \ , \ \forall \ \xi \in [-\epsilon_i^*, \epsilon_i] \Rightarrow \epsilon_i^* = \epsilon_i = (1-\rho)\epsilon \ . \tag{2.4.32}$$

Using (2.4.32) and conditions in Section 2.4.1 we get

$$\begin{aligned} c(\xi_i) = c(x_i, y_i, y_i + \epsilon_i + \xi_i) = c(x_i, y_i, y_i - \epsilon_i^* - \hat{\xi}_i) = c(\hat{\xi}_i) = \\ = \begin{cases} \frac{\xi_i^2}{4\rho\epsilon} \ , \ \xi_i \in [0, 2\rho\epsilon) \\ \xi_i - \rho\epsilon \ , \ \xi_i \in [2\rho\epsilon, \infty) \end{cases} . \end{aligned} \tag{2.4.33}$$

Thus, inserting (2.4.32) and (2.4.33) into (2.4.1) we arrive to the following formulation of the problem

$$\begin{aligned} \min_{\beta, \beta_0, \xi_i, \hat{\xi}_i} \quad & \frac{1}{2}||\beta||^2 + C \sum_{i=1}^{N} (c(\xi_i) + c(\hat{\xi}_i)) \\ \text{subject to} \quad & \xi_i, \hat{\xi}_i \geq 0, \ i = 1, ...N \\ & f(x_i) - y_i \leq (1-\rho)\epsilon + \xi_i, \ i = 1, ...N \\ & y_i - f(x_i) \leq (1-\rho)\epsilon + \hat{\xi}_i, \ i = 1, ...N \ . \end{aligned} \tag{2.4.34}$$

The Lagrangian for the primal problem corresponding to (2.4.34) is

$$\begin{aligned}\min_{\beta,\beta_0,\xi_i,\hat{\xi}_i} \quad & L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}(c(\xi_i) + c(\hat{\xi}_i)) \\ & - \sum_{i=1}^{N}\alpha_i(y_i - f(x_i) + (1-\rho)\epsilon + \xi_i) \\ & - \sum_{i=1}^{N}\alpha_i^*(f(x_i) - y_i + (1-\rho)\epsilon + \hat{\xi}_i) \\ & - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\hat{\xi}_i \\ \text{subject to} \quad & \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\ & \mu_i, \mu_i^* \geq 0,\ i = 1, ...N\ .\end{aligned} \tag{2.4.35}$$

Computing the derivatives in (2.4.35) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta + \sum_{i=1}^{N}\alpha_i h(x_i) - \sum_{i=1}^{N}\alpha_i^* h(x_i) = \beta + \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)h(x_i) \tag{2.4.36}$$

$$\frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) \tag{2.4.37}$$

$$\frac{\partial L_P}{\partial \xi_i} = C\frac{\partial c(\xi)}{\partial \xi} - \alpha_i - \mu_i,\ i = 1, ...N \tag{2.4.38}$$

$$\frac{\partial L_P}{\partial \hat{\xi}_i} = C\frac{\partial c(\hat{\xi})}{\partial \xi} - \alpha_i^* - \mu_i^*,\ i = 1, ...N\ , \tag{2.4.39}$$

and setting the derivatives (2.4.36), (2.4.37), (2.4.38) and (2.4.39) to zero we get:

$$\beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)h(x_i) \tag{2.4.40}$$

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \tag{2.4.41}$$

$$\alpha_i = C\frac{\partial c(\xi)}{\partial \xi} - \mu_i,\ i = 1, ...N \tag{2.4.42}$$

$$\alpha_i^* = C\frac{\partial c(\hat{\xi})}{\partial \xi} - \mu_i^*,\ i = 1, ...N\ . \tag{2.4.43}$$

Plugging (2.4.40), (2.4.41), (2.4.42) and (2.4.43) into (2.4.35) we get the dual problem, with the KKT conditions as constraints:

$$
\begin{aligned}
\max_{\alpha_i,\alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i, x_j) \\
& - (1-\rho)\epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) \\
& + C\sum_{i=1}^{N}(c(\xi_i) + c(\hat{\xi}_i) - \xi_i\frac{\partial c(\xi)}{\partial \xi} - \hat{\xi}_i\frac{\partial c(\hat{\xi})}{\partial \xi}) \\
\text{subject to} \quad & f(x_i) - y_i \leq (1-\rho)\epsilon + \xi_i,\ i = 1, ...N \\
& y_i - f(x_i) \leq (1-\rho)\epsilon + \hat{\xi}_i,\ i = 1, ...N \\
& \xi_i, \hat{\xi}_i \geq 0 \\
& \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\
& \alpha_i, \alpha_i^* \leq C\frac{\partial c(\xi)}{\partial \xi},\ i = 1, ...N \\
& \beta = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)h(x_i) \\
& \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \\
& \alpha_i(y_i - f(x_i) + (1-\rho)\epsilon + \xi_i) = 0 \\
& \alpha_i^*(f(x_i) - y_i + (1-\rho)\epsilon + \hat{\xi}_i) = 0 \\
& (C\frac{\partial c(\xi)}{\partial \xi} - \alpha_i)\xi_i = 0 \\
& (C\frac{\partial c(\hat{\xi})}{\partial \xi} - \alpha_i^*)\hat{\xi}_i = 0\ .
\end{aligned}
\tag{2.4.44}
$$

As shown in [28], the terms involving ξ_i and $\hat{\xi}_i$ can be simplified through easy steps, and the following equivalent dual form is obtained

$$
\begin{aligned}
\max_{\alpha_i,\alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i, x_j) \\
& - (1-\rho)\epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) \\
& - \frac{\rho\epsilon}{C}\sum_{i=1}^{N}(\alpha_i^2 + \alpha_i^{*2})\ ,
\end{aligned}
\tag{2.4.45}
$$

where the constraints are the same as in (2.4.44) with the exception of the Lagrangian coefficients α_i and α_i^* now being upper bounded by the hyperparameter C, i.e.:

$$
\alpha_i, \alpha_i^* \leq C\ \ i = 1, ...N\ . \tag{2.4.46}
$$

2. Zero-mean Laplace

The loss function here is

$$c(x_i, y_i, f(x_i)) = \frac{|f(x_i) - y_i|}{\sigma} \ . \tag{2.4.47}$$

Using the conditions detailed in Section 2.4.1 we get

$$\begin{aligned} & c(xi, y_i, y_i + \xi) = 0 \ , \ \forall \ \xi \in [-\epsilon_i^*, \epsilon_i] \Rightarrow \epsilon_i^* = \epsilon_i = 0 \Rightarrow \\ \Rightarrow & \begin{cases} c(\xi_i) = c(x_i, y_i, y_i + \epsilon_i + \xi_i) = \frac{|y_i + \epsilon_i + \xi_i - y_i|}{\sigma} = \frac{|\xi_i|}{\sigma} = \frac{\xi_i}{\sigma} \\ c(\hat{\xi}_i) = c(x_i, y_i, y_i - \epsilon_i^* - \hat{\xi}_i) = \frac{|y_i - \epsilon_i^* - \hat{\xi}_i - y_i|}{\sigma} = \frac{|-\hat{\xi}_i|}{\sigma} = \frac{\hat{\xi}_i}{\sigma} \end{cases} . \end{aligned} \tag{2.4.48}$$

Thus, inserting (2.4.48) into (2.4.1) we arrive to the following formulation of the problem

$$\begin{aligned} \min_{\beta, \beta_0, \xi_i, \hat{\xi}_i} \quad & \frac{1}{2}||\beta||^2 + \frac{C}{\sigma} \sum_{i=1}^{N} (\xi_i + \hat{\xi}_i) \\ \text{subject to} \quad & \xi_i, \hat{\xi}_i \geq 0, \ i = 1, ...N \\ & f(x_i) - y_i \leq \xi_i, \ i = 1, ...N \\ & y_i - f(x_i) \leq \hat{\xi}_i, \ i = 1, ...N \ . \end{aligned} \tag{2.4.49}$$

The Lagrangian for the primal problem corresponding to (2.4.49) is

$$\begin{aligned} \min_{\beta, \beta_0, \xi_i, \hat{\xi}_i} \quad & L_P = \frac{1}{2}||\beta||^2 + \frac{C}{\sigma} \sum_{i=1}^{N} (\xi_i + \hat{\xi}_i) \\ & - \sum_{i=1}^{N} \alpha_i (y_i - f(x_i) + \xi_i) \\ & - \sum_{i=1}^{N} \alpha_i^* (f(x_i) - y_i + \hat{\xi}_i) \\ & - \sum_{i=1}^{N} \mu_i \xi_i - \sum_{i=1}^{N} \mu_i^* \hat{\xi}_i \\ \text{subject to} \quad & \alpha_i, \alpha_i^* \geq 0, \ i = 1, ...N \\ & \mu_i, \mu_i^* \geq 0, \ i = 1, ...N \ . \end{aligned} \tag{2.4.50}$$

Computing the derivatives in (2.4.50) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta + \sum_{i=1}^{N} \alpha_i h(x_i) - \sum_{i=1}^{N} \alpha_i^* h(x_i) = \beta + \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) h(x_i) \tag{2.4.51}$$

$$\frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \tag{2.4.52}$$

$$\frac{\partial L_P}{\partial \xi_i} = \frac{C}{\sigma} - \alpha_i - \mu_i, \ i = 1, ...N \tag{2.4.53}$$

$$\frac{\partial L_P}{\partial \hat{\xi}_i} = \frac{C}{\sigma} - \alpha_i^* - \mu_i^*,\ i = 1, ...N\ , \tag{2.4.54}$$

and setting the derivatives (2.4.51), (2.4.52), (2.4.53) and (2.4.54) to zero we get:

$$\beta = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) h(x_i) \tag{2.4.55}$$

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0 \tag{2.4.56}$$

$$\alpha_i = \frac{C}{\sigma} - \mu_i,\ i = 1, ...N \tag{2.4.57}$$

$$\alpha_i^* = \frac{C}{\sigma} - \mu_i^*,\ i = 1, ...N\ . \tag{2.4.58}$$

Plugging (2.4.55), (2.4.56), (2.4.57) and (2.4.58) into (2.4.50) we get the dual problem, with the KKT conditions as constraints:

$$\begin{aligned}
\max_{\alpha_i, \alpha_i^*} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i, x_j) \\
\text{subject to} \quad & f(x_i) - y_i \leq \xi_i,\ i = 1, ...N \\
& y_i - f(x_i) \leq \hat{\xi}_i,\ i = 1, ...N \\
& \xi_i, \hat{\xi}_i \geq 0 \\
& \alpha_i, \alpha_i^* \geq 0,\ i = 1, ...N \\
& \alpha_i, \alpha_i^* \leq \frac{C}{\sigma},\ i = 1, ...N \\
& \beta = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) h(x_i) \\
& \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0 \\
& \alpha_i (y_i - f(x_i) + \xi_i) = 0 \\
& \alpha_i^* (f(x_i) - y_i + \hat{\xi}_i) = 0 \\
& (\frac{C}{\sigma} - \alpha_i) \xi_i = 0 \\
& (\frac{C}{\sigma} - \alpha_i^*) \hat{\xi}_i = 0\ .
\end{aligned} \tag{2.4.59}$$

Following an analogous procedure, we can get the dual problem formulation for the non-zero-mean Laplace loss function choice.

3. Zero-mean Gaussian

The loss function for this case is

$$c(x_i, y_i, f(x_i)) = \frac{(f(x_i) - y_i)^2}{2\sigma^2} \ . \tag{2.4.60}$$

Following the approach proposed in [36] for the gaussian case, we use a slightly different formulation of (2.4.1), changing the slack variables to

$$\xi_i = y_i - f(x_i) - \epsilon \ . \tag{2.4.61}$$

This allows for negative slack values, so it is not necessary to add variables $\hat{\xi}_i$. Thus, the problem in (2.4.1) is reformulated as

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i} \quad & \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N} c(\xi_i) \\ \text{subject to} \quad & y_i - f(x_i) = \xi_i, \ i = 1, ...N \ . \end{aligned} \tag{2.4.62}$$

Using the conditions detailed in Section 2.4.1 we get

$$\begin{aligned} c(xi, y_i, y_i + \xi) = 0 \ , \ \forall \ \xi \in [-\epsilon_i^*, \epsilon_i] \Rightarrow \epsilon_i^* = \epsilon_i = 0 \Rightarrow \\ \Rightarrow c(\xi_i) = c(x_i, y_i, y_i + \epsilon_i + \xi_i) = \frac{(y_i + \epsilon_i + \xi_i - y_i)^2}{2\sigma^2} = \frac{\xi_i^2}{2\sigma^2} \ . \end{aligned} \tag{2.4.63}$$

Thus, inserting (2.4.63) into (2.4.62) we arrive to the following formulation of the problem

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i} \quad & \frac{1}{2}||\beta||^2 + \frac{C}{2\sigma^2}\sum_{i=1}^{N} \xi_i^2 \\ \text{subject to} \quad & y_i - f(x_i) = \xi_i, \ i = 1, ...N \ . \end{aligned} \tag{2.4.64}$$

The Lagrangian for the primal problem corresponding to (2.4.64) is

$$\begin{aligned} \min_{\beta,\beta_0,\xi_i,\hat{\xi}_i} \quad & L_P = \frac{1}{2}||\beta||^2 + \frac{C}{2\sigma^2}\sum_{i=1}^{N} \xi_i^2 \\ & - \sum_{i=1}^{N} \alpha_i (f(x_i) - y_i + \xi_i) \\ \text{subject to} \quad & \alpha_i \geq 0, \ i = 1, ...N \ . \end{aligned} \tag{2.4.65}$$

Computing the derivatives in (2.4.65) we obtain

$$\frac{\partial L_P}{\partial \beta} = \beta - \sum_{i=1}^{N} \alpha_i h(x_i) \tag{2.4.66}$$

$$\frac{\partial L_P}{\partial \beta_0} = -\sum_{i=1}^{N} \alpha_i \tag{2.4.67}$$

$$\frac{\partial L_P}{\partial \xi_i} = \frac{C}{\sigma^2}\xi_i - \alpha_i,\ i = 1, ...N\ , \tag{2.4.68}$$

and setting the derivatives (2.4.66), (2.4.67) and (2.4.68) to zero we get:

$$\beta = \sum_{i=1}^{N} \alpha_i h(x_i) \tag{2.4.69}$$

$$\sum_{i=1}^{N} \alpha_i = 0 \tag{2.4.70}$$

$$\alpha_i = \frac{C}{\sigma^2}\xi_i,\ i = 1, ...N\ . \tag{2.4.71}$$

Plugging (2.4.69), (2.4.70) and (2.4.71) into (2.4.65) we get the dual problem, with the KKT conditions as constraints:

$$\begin{aligned}
\max_{\alpha_i} \quad & L_D = \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(K(x_i, x_j) + \frac{\delta_{ij}\sigma^2}{C}) \\
\text{subject to} \quad & y_i - f(x_i) = \xi_i,\ i = 1, ...N \\
& \alpha_i \geq 0,\ i = 1, ...N \\
& \beta = \sum_{i=1}^{N} \alpha_i h(x_i) \\
& \sum_{i=1}^{N} \alpha_i = 0 \\
& \alpha_i = \frac{C}{\sigma^2}\xi_i,\ i = 1, ...N \\
& \alpha_i(f(x_i) - y_i + \xi_i) = 0\ .
\end{aligned} \tag{2.4.72}$$

Following an analogous procedure, we can get the dual problem formulation for the non-zero-mean Gaussian loss function choice.

4. Beta

In this case, the loss function is

$$c(\xi_i) = (1 - \alpha)\log(\xi_i) + (1 - \beta)\log(1 - \xi_i)\ . \tag{2.4.73}$$

For conciseness, for this case we will plug (2.4.73) directly into (2.4.28). For the beta distribution we have

$$\frac{\partial c(\xi)}{\partial \xi} = \frac{1 - \alpha}{\xi_i} - \frac{1 - \beta}{1 - \xi_i}\ . \tag{2.4.74}$$

Replacing (2.4.73) and (2.4.74) into equations (2.4.29) and (2.4.30) we get

$$\begin{aligned} T_i(\xi_i) &= c(\xi_i) - \xi_i \frac{\partial c(\xi)}{\partial \xi} = \\ &= (1-\alpha)\log(\xi_i) + (1-\beta)\log(1-\xi_i) - \xi_i\left(\frac{1-\alpha}{\xi_i} - \frac{1-\beta}{1-\xi_i}\right) \end{aligned} \tag{2.4.75}$$

$$\begin{aligned} T_i^*(\xi_i) &= c(\hat{\xi}_i) - \hat{\xi}_i \frac{\partial c(\hat{\xi})}{\partial \xi} = \\ &= (1-\alpha)\log(\hat{\xi}_i) + (1-\beta)\log(1-\hat{\xi}_i) - \hat{\xi}_i\left(\frac{1-\alpha}{\hat{\xi}_i} - \frac{1-\beta}{1-\hat{\xi}_i}\right) . \end{aligned} \tag{2.4.76}$$

Finally, plugging (2.4.75) and (2.4.76) into (2.4.28) we can obtain the dual formulation for the beta error asumption case.

5. Weibull

In this case, the loss function is

$$c(\xi_i) = \begin{cases} (1-\kappa)\log(\xi_i) + (\frac{\xi_i}{\lambda})^{\kappa} \ , \ \xi_i > 0 \\ 0 \ , \ \xi_i \leq 0 \end{cases} . \tag{2.4.77}$$

As in the beta case, for conciseness we will plug (2.4.77) directly into (2.4.28). For the weibull distribution we have

$$\frac{\partial c(\xi)}{\partial \xi} = \begin{cases} \frac{1-\kappa}{\xi_i} + \frac{\kappa}{\lambda}(\frac{\xi_i}{\lambda})^{\kappa-1} \ , \ \xi_i > 0 \\ 0 \ , \ \xi_i \leq 0 \end{cases} . \tag{2.4.78}$$

Replacing (2.4.77) and (2.4.78) into equations (2.4.29) and (2.4.30) we get

$$T_i(\xi_i) = c(\xi_i) - \xi_i \frac{\partial c(\xi)}{\partial \xi} = \begin{cases} (1-\kappa)\log\xi_i + (\frac{\xi_i}{\lambda})^{\kappa} - \xi_i(\frac{1-\kappa}{\xi_i} + \frac{\kappa}{\lambda}(\frac{\xi_i}{\lambda})^{\kappa-1}) \ , \ \xi_i > 0 \\ 0 \ , \ \xi_i \leq 0 \end{cases} \tag{2.4.79}$$

$$T_i^*(\xi_i) = c(\hat{\xi}_i) - \hat{\xi}_i \frac{\partial c(\hat{\xi})}{\partial \xi} = \begin{cases} (1-\kappa)\log\hat{\xi}_i + (\frac{\hat{\xi}_i}{\lambda})^{\kappa} - \hat{\xi}_i(\frac{1-\kappa}{\hat{\xi}_i} + \frac{\kappa}{\lambda}(\frac{\hat{\xi}_i}{\lambda})^{\kappa-1}) \ , \ \xi_i > 0 \\ 0 \ , \ \xi_i \leq 0 \end{cases} , \tag{2.4.80}$$

Finally, plugging (2.4.79) and (2.4.80) into (2.4.28) we can obtain the dual formulation for the Weibull error asumption case.

Recall that this methodology can only be applied when the objective function to minimize is convex. If this is not the case, a proxy loss function or non-convex optimization method must be applied.

Chapter 3

Uncertainty Estimates

Support vector regression, SVR, [37] has been widely used in regression problems such as stock market [14], wind energy [15] and solar radiation [16] forecasting. Classical SVR, however, does not give probability intervals to address the uncertainty in the predictions and, in fact, error interval estimation for SVR has received a somewhat limited attention in the literature. Notice that here approaches such as the well known ones for linear regression under Gaussian models completely break down, not only by the difficulty of ensuring normal random variables but, above all, by the fact that the familiar analytic estimates of the linear coefficients are simply impossible in SVR and, of course, less so any asymptotic analysis.

In [28], a Bayesian interpretation of SVR is described and then used to propose methods to, first, determine SVR parameters by maximizing an evidence function and, second, derive probability intervals for predictions. A drawback of these methods is that they modify the classical SVR formulation of the problem to solve, and hence existing SVR software, such as the popular LIBSVM [38], cannot be used, at least without modifying it first.

A more direct approach is proposed in [32], which assumes prediction errors to follow a specific probability distribution that, in turn, is used to define probability intervals for them. Zero-mean Gaussian and Laplace families are proposed as noise models and fitted by maximum likelihood estimation, MLE, [39] using out-of-sample residuals of SVR models; optimal SVR parameters are obtained simply by cross validation. In this thesis, we follow this methodology to give probability intervals under the assumption of both zero-mean Laplace and Gaussian distributions, as well as for their non–zero mean counterparts plus the Beta and Weibull distributions.

As described before, general error models for SVR other than the well known ϵ–insensitive loss have been proposed in [30]. This suggests that noise distribution might be different across particular problems and it should be reflected in the particular SVR model to be used. If the assumption is true and the underlying noise distribution is accurately estimated, one should expect a reduction in interval prediction errors. We study if the proposed method can estimate this noise distribution. To this end, we detail in this chapter a test for testing distribution hypotheses to be used as accuracy benchmark.

A difficulty with the method proposed in [32] is that it assumes the residual distribution to be independent of the predicted value and, therefore, probability intervals have exactly

the same width for all input instances. To lessen the impact of this drawback, we propose to use clustering methods to split the data into several groups and build different intervals for each one of them. In this chapter we describe k-means, one of the standard clustering algorithms.

3.1 Bayesian SVR

In Section 2.3 we described Bayesian interpretations of SVR. Here we show to obtain error intervals for these models.

The prediction error interval for $\delta_i = y_i - \hat{f}(x_i)$ is $(-p_s, p_s)$, where p_s is the upper sth percentile of the corresponding probability distribution of $\Psi(= y - \hat{f}(x)))$ and $\hat{f} = (\alpha - \alpha^*)^T K$ is the solution of (2.3.18).

As shown in [28], given a pre-specified probability $1 - 2s$, if we denote p_Ψ as the density of Ψ we have

$$1 - s = \int_{-\infty}^{p_s} p_\Psi(z)dz = \int_{-\infty}^{+\infty} \int_{-\infty}^{p_s - f} p(\delta)d\delta p_{f|D}(f)df \tag{3.1.1}$$

$$p(\delta) = \frac{\exp\left(-C \cdot l_{\beta,\epsilon}(\delta)\right)}{\int \exp\left(-C \cdot l_{\beta,\epsilon}(\delta)\right)d\delta},\ p(f(x)|D) = \frac{1}{\sqrt{2\pi}\sigma_t} \exp\left(-\frac{(f(x) - \hat{f})^2}{2\sigma_t^2}\right), \tag{3.1.2}$$

where $\sigma_t^2 = K(x,x) - K_{F,x}^T K_{F,F}^{-1} K_{F,x}$ and $K_{F,x}$ the vector containing all $K(x_i, x)$, with $i \in F = \{i | 0 < \alpha_i < C \text{ or } 0 < \alpha_i^* < C\}$ and $K_{F,F}$ the corresponding submatrix.

Here, the distribution of Ψ depends on the input value x, and so does p_s . Thus, the numerical integration needed to solve (3.1.1) must be carried out for each test instance.

3.2 Proposed Approach

In [32] a different and simpler way of obtaining probability estimates for SVR is presented. They propose to model the distribution of Ψ based on a set of out-of-sample residuals $\{\psi_i\}_{i=1}^l$. These residuals are the result of conducting a k-fold cross-validation over the training data D to get the estimated function $\hat{f}_j, j = 1, ..., k$, and then setting $\psi_i \equiv \hat{f}_j(x_i) - y_i$ for (x_i, y_i) in the jth fold of the training data.

In this research the authors assume that the conditional distribution of y given x depends on x only through $\hat{f}(x)$. In theory, the distribution of Ψ may depend on the input x, and therefore the length of the predictive interval with a pre-specified coverage probability may vary from one example to another, reflecting the fact that the prediction variances vary with different input values. However, they claim that despite the fact that their interval Ψ is not influenced by x, so does not reflect this property, it can be justified if we consider the probability to be taken over all possible input values. To reduce the loss of accuracy that this assumption can provoke when working with data whose distribution strongly depends on variables, we propose to cluster data into different groups and apply the proposed technique on each group.

The authors propose to model ψ_i by zero-mean Gaussian and Laplace distributions because residuals of data studied in their work seem to be simmetric about zero and both Gaussian and Laplace captured their shape reasonably well. Here we describe how to model ψ_i using these distributions following the method proposed in [32] and add to them four more: Gaussian and Laplace distributions having mean not zero, Beta distribution and Weibull distribution. We choose these distributions because previous works, such as [29] for Beta distribution and [31] for Weibull distribution, have shown their usefulness to model real-world problems such as wind speed and wind power production.

For Laplace and Gaussian distributions, the ψ_i's are generated by cross-validation over the training set or using a fixed validation set . For the beta distribution, values are scaled so $\psi_i \in (0, 1)$ and for the Weibull distribution residuals are set to $\psi_i \equiv |\hat{f}(x_i) - y_i|$

3.2.1 Parameter Estimation

Assuming that ψ_i are independent, we can estimate the distributions parameters by maximizing the likelihood L. If ψ_i are independent we have $L(\theta; \psi_1 ... \psi_n) = \prod_{i=1}^{n} f(\psi_i|\theta)$ and, denoting l the logarithm of the likelihood, $l(\theta; \psi_1 ... \psi_n) = \sum_{i=1}^{n} \log f(\psi_i|\theta)$, where f represents the density function of the distribution of ψ_i. Maximizing l is equivalent to maximize L.

1. Zero Mean Laplace

$$l(\theta; \psi_1 ... \psi_n) = \sum_{i=1}^{n} \log \frac{1}{2\sigma} e^{-\frac{|\psi_i|}{\sigma}} =$$

$$= \sum_{i=1}^{n} \log \frac{1}{2\sigma} - \sum_{i=1}^{n} \frac{|\psi_i|}{\sigma} = -n \log 2 - n \log \sigma - \frac{1}{\sigma} \sum_{i=1}^{n} |\psi_i| \ . \tag{3.2.1}$$

Now, in the maximum the first derivative must be zero, so:

$$\frac{\partial l}{\partial \hat{\sigma}} = -n \frac{1}{\hat{\sigma}} + \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} |\psi_i| = 0 \ . \tag{3.2.2}$$

Solving (3.2.2) we obtain:

$$\hat{\sigma} = \frac{\sum_{i=1}^{n} |\psi_i|}{n} \ , \tag{3.2.3}$$

i.e. the mean absolute error, MAE.

2. Non-zero Mean Laplace

$$l(\theta; \psi_1 ... \psi_n) = \sum_{i=1}^{n} \log \frac{1}{2\sigma} e^{-\frac{|\psi_i - \mu|}{\sigma}}$$

Following the same steps as for the zero-mean Laplace we get:

$$\hat{\sigma} = \frac{\sum_{i=1}^{n} |\psi_i - \mu|}{n} \ . \tag{3.2.4}$$

And we set the mean to be:

$$\mu = m_{\delta_i} \ , \tag{3.2.5}$$

where m_{δ_i} denotes the median of the δ_i residuals

3. Zero Mean Gaussian

$$l(\theta; \psi_1 ... \psi_n) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\psi_i^2}{2\sigma^2}} =$$

$$= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_{i=1}^{n} \frac{\psi_i^2}{2\sigma^2} = -n \log \sqrt{2\pi} - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \psi_i^2 \ . \tag{3.2.6}$$

Setting the derivative to zero:

$$\frac{\partial l}{\partial \hat{\sigma}} = -n \frac{1}{\hat{\sigma}} + \frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} \psi_i^2 = 0 \ . \tag{3.2.7}$$

Solving (3.2.7) we obtain:

$$\hat{\sigma} = \frac{\sum_{i=1}^{n} \psi_i^2}{n} \ , \tag{3.2.8}$$

i.e. the mean squared error, MSE.

4. Non-zero Mean Gaussian

$$l(\theta; \psi_1 ... \psi_n) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\psi_i - \mu)^2}{2\sigma^2}}$$

Following the same steps as for the zero-mean Gaussian we get:

$$\hat{\sigma} = \frac{\sum_{i=1}^{n} (\psi_i - \mu)^2}{n} \ , \tag{3.2.9}$$

and we set the mean to be:

$$\mu = \sum_{i=1}^{n} \frac{\psi_i}{n} \ . \tag{3.2.10}$$

5. Beta

$$l(\theta;\psi_1...\psi_n)=\sum_{i=1}^{n}\log\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\psi_i^{\alpha-1}(1-\psi_i)^{\beta-1}=$$

$$=\sum_{i=1}^{n}\log\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}+\sum_{i=1}^{n}\log\psi_i^{\alpha-1}+\sum_{i=1}^{n}\log(1-\psi_i)^{\beta-1}=$$

$$=n(\log\Gamma(\alpha+\beta)-\log\Gamma(\alpha)-\log\Gamma(\beta))+(\alpha-1)\sum_{i=1}^{n}\log\psi_i+(\beta-1)\sum_{i=1}^{n}\log(1-\psi_i)\ . \tag{3.2.11}$$

Setting the derivatives to zero we get:

$$\frac{\partial l}{\partial\hat{\alpha}}=n(\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)}-\frac{\Gamma'(\alpha)}{\Gamma(\alpha)})+\sum_{i=1}^{n}\log\psi_i=0 \tag{3.2.12}$$

$$\frac{\partial l}{\partial\hat{\beta}}=n(\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)}-\frac{\Gamma'(\beta)}{\Gamma(\beta)})+\sum_{i=1}^{n}\log(1-\psi_i)=0\ . \tag{3.2.13}$$

Denoting $\phi(x)=\frac{\Gamma'(x)}{\Gamma(x)}$ in (3.2.13) we end with the following system of two equations:

$$\frac{\sum_{i=1}^{n}\log\psi_i}{n}=\phi(\alpha)-\phi(\alpha+\beta)$$

$$\frac{\sum_{i=1}^{n}\log(1-\psi_i)}{n}=\phi(\beta)-\phi(\alpha+\beta)\ . \tag{3.2.14}$$

The expressions on the right-hand sides of the equations in (3.2.14) may be denoted $F_1(\alpha,\beta)$ and $F_2(\alpha,\beta)$ respectively, and the equations rewritten as:

$$\frac{\sum_{i=1}^{n}\log\psi_i}{n}=F_1(\alpha,\beta)$$

$$\frac{\sum_{i=1}^{n}\log(1-\psi_i)}{n}=F_2(\alpha,\beta)\ . \tag{3.2.15}$$

Iterative methods may be employed for the numerical solution of the equations (3.2.15). Newton-Raphson's method, involving the linearization of F_1 and F_2 in the neighborhood of the root, leads to the following iterative scheme:

$$\frac{\sum_{i=1}^{n}\log\psi_i}{n}=F_1(\alpha_j,\beta_j)+(\alpha_{j+1}-\alpha_j)\left(\frac{\partial F_1}{\partial\alpha}\right)_{(\alpha_j,\beta_j)}+(\beta_{j+1}-\beta_j)\left(\frac{\partial F_1}{\partial\beta}\right)_{(\alpha_j,\beta_j)}$$

$$\frac{\sum_{i=1}^{n}\log(1-\psi_i)}{n}=F_2(\alpha_j,\beta_j)+(\alpha_{j+1}-\alpha_j)\left(\frac{\partial F_2}{\partial\alpha}\right)_{(\alpha_j,\beta_j)}+(\beta_{j+1}-\beta_j)\left(\frac{\partial F_2}{\partial\beta}\right)_{(\alpha_j,\beta_j)}\ . \tag{3.2.16}$$

The initial values (α_0, β_0) are pivotal for the efficient convergence of Newton-Raphson's method. We use the values proposed in [40], obtained using moment estimates:

$$\alpha_0 = \frac{m_1(m_1 - m_2)}{m_2 - m_1^2},\ \beta_0 = \frac{\alpha_0(1 - m_1)}{m_1}\ . \tag{3.2.17}$$

6. Weibull [1]

$$l(\theta; \psi_1 ... \psi_n) = \sum_{i=1}^{n} \log \left(\frac{\kappa}{\lambda} \left(\frac{\psi_i}{\lambda} \right)^{\kappa - 1} e^{-\left(\frac{\psi_i}{\lambda}\right)^{\kappa}} \right) =$$

$$= \sum_{i=1}^{n} \log \frac{k}{\lambda} + \sum_{i=1}^{n} \log \left(\frac{\psi_i}{\lambda} \right)^{\kappa - 1} - \sum_{i=1}^{n} \left(\frac{\psi_i}{\lambda} \right)^{\kappa} =$$

$$= n \log \kappa - n \log \lambda + (\kappa - 1) n \sum_{i=1}^{n} \log \psi_i - (\kappa - 1) n \log \lambda - \frac{1}{\lambda^k} \sum_{i=1}^{n} \psi_i^k\ . \tag{3.2.18}$$

Setting the derivatives to zero we get:

$$\frac{\partial l}{\partial \hat{\lambda}} = -\frac{n}{\lambda} - \frac{\kappa n - n}{\lambda} + \frac{\kappa}{\lambda^{\kappa+1}} \sum_{i=1}^{n} \psi_i^{\kappa} = 0 \tag{3.2.19}$$

$$\frac{\partial l}{\partial \hat{\kappa}} = \frac{n}{\kappa} + n \sum_{i=1}^{n} \log \psi_i - n \log \lambda - \sum_{i=1}^{n} \left(\frac{\psi_i}{\lambda} \right)^{\kappa} \log \frac{\psi_i}{\lambda} = 0\ . \tag{3.2.20}$$

Solving (3.2.19) we obtain:

$$\lambda = \left(\frac{1}{n} \sum_{i=1}^{n} \psi_i^{\kappa} \right)^{\frac{1}{\kappa}}\ . \tag{3.2.21}$$

Putting (3.2.21) into (3.2.20) we get:

$$\frac{\sum_{i=1}^{n} \log \psi_i}{n} = \frac{\sum_{i=1}^{n} \psi_i^{\kappa} \log \psi_i}{\sum_{i=1}^{n} \psi_i^{\kappa}} - \frac{1}{\kappa}\ . \tag{3.2.22}$$

Denoting the expression on the right-hand side of (3.2.22) as $G(\kappa)$ the equation can be rewritten as:

$$\frac{\sum_{i=1}^{n} \log \psi_i}{n} = G(\kappa)\ . \tag{3.2.23}$$

[1] We only consider the case $z \geq 0$ because Weibull distribution is zero for negative values, so they do not affect the log-likelihood function.

The existence and uniqueness of the solution of (3.2.23) can be proved. A simple proof is provided in Appendix A. As in the beta case, we use Newton-Raphson's method to solve (3.2.23), obtaining the following iterative scheme:

$$\frac{\sum_{i=1}^{n} \log \psi_i}{n} = G(\kappa_j) + (\kappa_{j+1} - \kappa_j) \left(\frac{\partial G}{\partial \kappa} \right)_{(\kappa_j)} . \tag{3.2.24}$$

This time the initial value κ_0 is chosen empirically through experimentation. In our case $\kappa_0 = 1$ seems to ensure a fast convergence.

3.2.2 Probability Intervals

Given a pre-specified probability 1 - 2s, we want to obtain the prediction error interval, (a, b), for each instance (x_i, y_i) in the test set. As stated before, the conditional distribution of y given x is assumed to depend on x only through the prediction value $\hat{f}(x)$ and therefore the value of this interval is the same for each test instance.

1. **Zero Mean Laplace and Gaussian:** We need to compute the upper sth percentile, p_s, of the corresponding probability distribution of $\Psi(= \hat{f}(x) - y)$. For a zero-mean symmetric variable with density $p(z)$, we can obtain p_s just by solving

$$1 - s = \int_{-\infty}^{p_s} p(z)dz . \tag{3.2.25}$$

   The prediction error interval is $(-p_s, p_s)$ in this case.

2. **Non-Zero Mean Laplace and Gaussian:** For a non-zero mean Laplace or Gaussian distribution the percentile p_s is determined as before:

$$1 - s = \int_{-\infty}^{p_s} p(z)dz . \tag{3.2.26}$$

   However, as in this case the distribution is centered at μ and not zero, the prediction error interval is $(\mu - (p_s - \mu), \mu + (p_s - \mu))$.

3. **Beta:** For a beta distribution $z \geq 0$, so we obtain p_s by solving

$$1 - s = \int_{0}^{p_s} p(z)dz . \tag{3.2.27}$$

   The prediction error interval is $(0, p_s)$.

4. **Weibull:** As stated before, for the Weibull distribution we only consider the case $z \geq 0$, so we determine the prediction error interval the same way as for the beta distribution.

3.3 Test for Testing Distribution Hypotheses

As a way of validating the proposed approach usefulness as a mechanism to detect the distribution of errors, we will compare in Section 4 its results against the ones obtained using a test for testing hypotheses about the distribution of random variables.

The test we use is described in [41], where it is stated that if $Z_1, Z_2, ..., Z_t$ are a random sample from a distribution with density

$$\frac{1}{\sigma^t} p(\frac{z_1}{\sigma}) p(\frac{z_2}{\sigma}) ... p(\frac{z_t}{\sigma}) \ , \tag{3.3.1}$$

we can define the statistic

$$T(p_0, p_1, \{z_i\}_{i=1}^t) = \frac{\int_0^\infty \tau^{t-1} p_1(\tau z_1) p_1(\tau z_2) ... p_1(\tau z_t) d\tau}{\int_0^\infty \tau^{t-1} p_0(\tau z_1) p_0(\tau z_2) ... p_0(\tau z_t) d\tau} \ . \tag{3.3.2}$$

The most powerful test among all tests which are invariant under scale transformation for testing hypothesis $H_0 : p = p_0$ against hypothesis $H_1 : p = p_1$ rejects H_0 when

$$T(p_0, p_1, \{z_i\}_{i=1}^t) > c_\alpha \ , \tag{3.3.3}$$

where c_α is a threshold associated to a given significance level α. At a certain value of c_α, when H_0 is true the probability of rejecting this hypothesis is α, i.e., c_α is determined so that the probability of rejecting H_0 is α when H_0 is actually true. Thus, it holds that

$$P_0(T(p_0, p_1, \{z_i\}_{i=1}^t) > c_\alpha) = \alpha \ , \tag{3.3.4}$$

where P_0 is the probability under H_0 and typically, the standard value $\alpha = 0.05$ is the one used as significance level.

We solve (3.3.2) by numerical integration. See implementation details in 4.3.

3.4 Clustering. *K*-means

As explained before, the proposed approach to build intervals takes the assumption that the prediction error interval is not influenced by the input values $\{x_i\}_{i=1}^N$, so the interval is constant for all instances in the test dataset. To try to limit the loss of accuracy that this assumption can cause, we propose to cluster data into different groups and apply the proposed technique on each group. Two approaches will be followed:

- Use of empirical clusterings based on prior knowledge of the particular prediction problem analyzed.

- Use of standard clustering techniques. K-means algorithm will be our choice.

Empirical techniques designed specifically for a particular problem will be described in section 4.1. Here we describe the K-means algorithm.

3.4.1 *K*-means Goal and Initialization Methods

In [42] and [43], the K-means algorithm is proposed. His aim is to divide M points in N dimensions into K clusters so that the within-cluster sum of squares is minimized. In other words, its objective is to find:

$$\min_{S} \quad \sum_{i=1}^{K} \sum_{x \in s_i} ||x - C_i||^2 \ , \tag{3.4.1}$$

where $S = \{s_1, s_2, ..., s_K\}$ are the different clusters created and C_i is the centroid of s_i.

The algorithm requires as input a matrix of M points in N dimensions and a matrix of K initial cluster centroids in N dimensions, $C^0 = \{C_1, C_2, ..., C_K\}$. The solution found by K-means depends on the choice of C^0, i.e. this is not an unique solution problem. Commonly used initialization methods to choose C^0 are Forgy and Random Partition:

- **The Forgy method:** This method randomly chooses K observations from the data set and uses them as the initial cluster centroids.

- **The Random Partition:** Here, a cluster is assigned randomly to each observation and then the initial mean of the points allocated to each cluster is used as the centroid of the clusters.

As there is not an unique solution and the K-means algorithm can get stuck at a bad local minimum, it is recommendable to run the algorithm for different initial centroids values and choose the solution that gives a smaller within-cluster sum of squares among all executions.

The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to [44], the Random Partition method is generally preferable for algorithms such as the k-harmonic means and fuzzy K-means. For expectation maximization and standard K-means algorithms as the one used here, the Forgy method of initialization is preferable and will be the one applied in our experiments.

3.4.2 Iterative Steps and Convergence

Given an initial set of cluster centroids C^0, the K-means algorithm proceeds by iterating two steps:

- **Assignment step:** Assign each observation, x_i, to the cluster s_w with the minimum euclidean distance between its centroid, C_w, and the observation, i.e. x_i is assigned to cluster s_w where

$$\min_w \quad ||x_i - C_w|| \ . \tag{3.4.2}$$

- **Update step:** Compute the mean of all points in each cluster and set it to be the new cluster centroid.

$$C_i = \frac{1}{|s_i|} \sum_{x \in s_i} x \ , \ i = 1, ...K \ , \tag{3.4.3}$$

  where $|s_i|$ is the total number of points in cluster i.

These two steps are iterated until some criterion of convergence is reached. As a result of the previous loop, the K centroids may change their position in a step by step manner. Eventually, a situation will be reached where the centroids do not move anymore, i.e. $C^i = C^{i-1}$. This signifies the convergence criterion for clustering and hence at this point the iterations stop and the resulting clusters are the final solution given by the K-means algorithm. Thus, at least one iteration of the algorithm is needed to reach convergence, as it is necessary to observe no change in the centroids between the beginning of an iteration and its end for the converge criterion to be reached.

3.4.3 Selection of *K*. Algorithm's Goodness

In some cases, the choice of value for K, i.e. into how many clusters we are looking to split the data, is directly defined by the problem. For instance, we may want to split customers of a particular retailer into high-value clients and low-value ones, and therefore the value of K has to be 2 to solve this problem.

When this is not the case and there is no direct choice of K, some technique must be used to make this decision. How to automatically select the value of K, has been the subject of different studies and various methods have been suggested, as for instance the elbow method or elbow rule. [45] gives a review on these proposed methods.

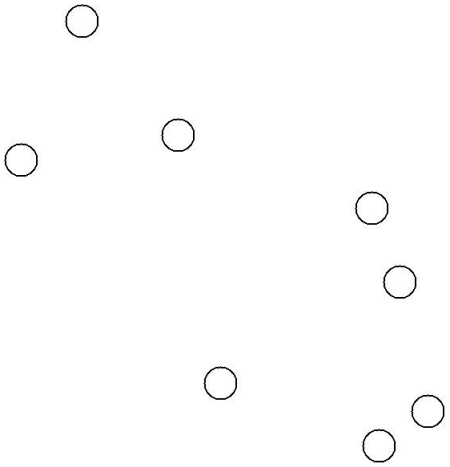
However, typically the K-means algorithm is employed as a previous step to a posterior application of other algorithm or model, as is the case in this work, where we use K-means prior to the use of our proposed approach to build prediction error intervals, aiming to improve their accuracy. In these cases, it may be preferable to select the value of K according to its positive or negative impact in this posterior goal. In our case, K is chosen as the value that produces the greatest improvement in the error intervals accuracy.

Algorithm 1: K-means algorithm.

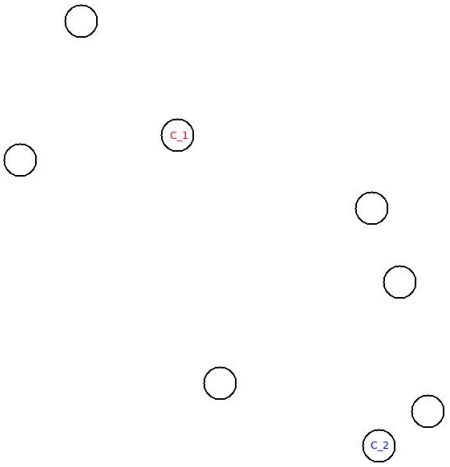
```
Initialization of centroids: We use the Forgy method
while C^i != C^{i-1} do
    Assignment step
    Update step
end
return S = {s_1, s_2, ..., s_K}
```

3.4.4 Algorithm Summary

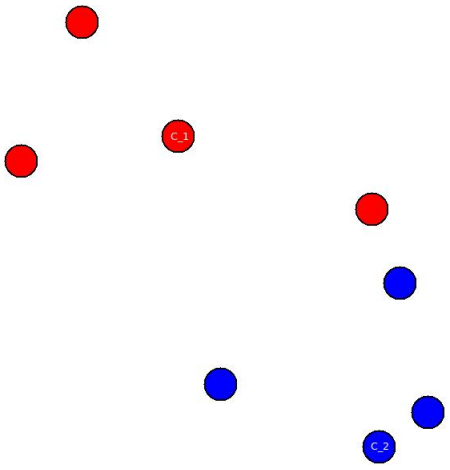
Gathering together all steps earlier described, the K-means algorithm can be summarized as Algorithm 1. A visualization of a K-means algorithm execution is shown in the following figures.



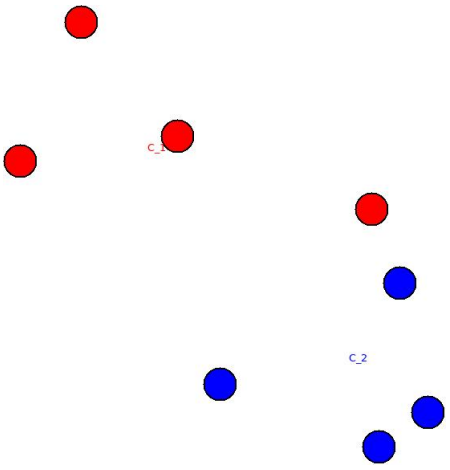
(a) Initial data.



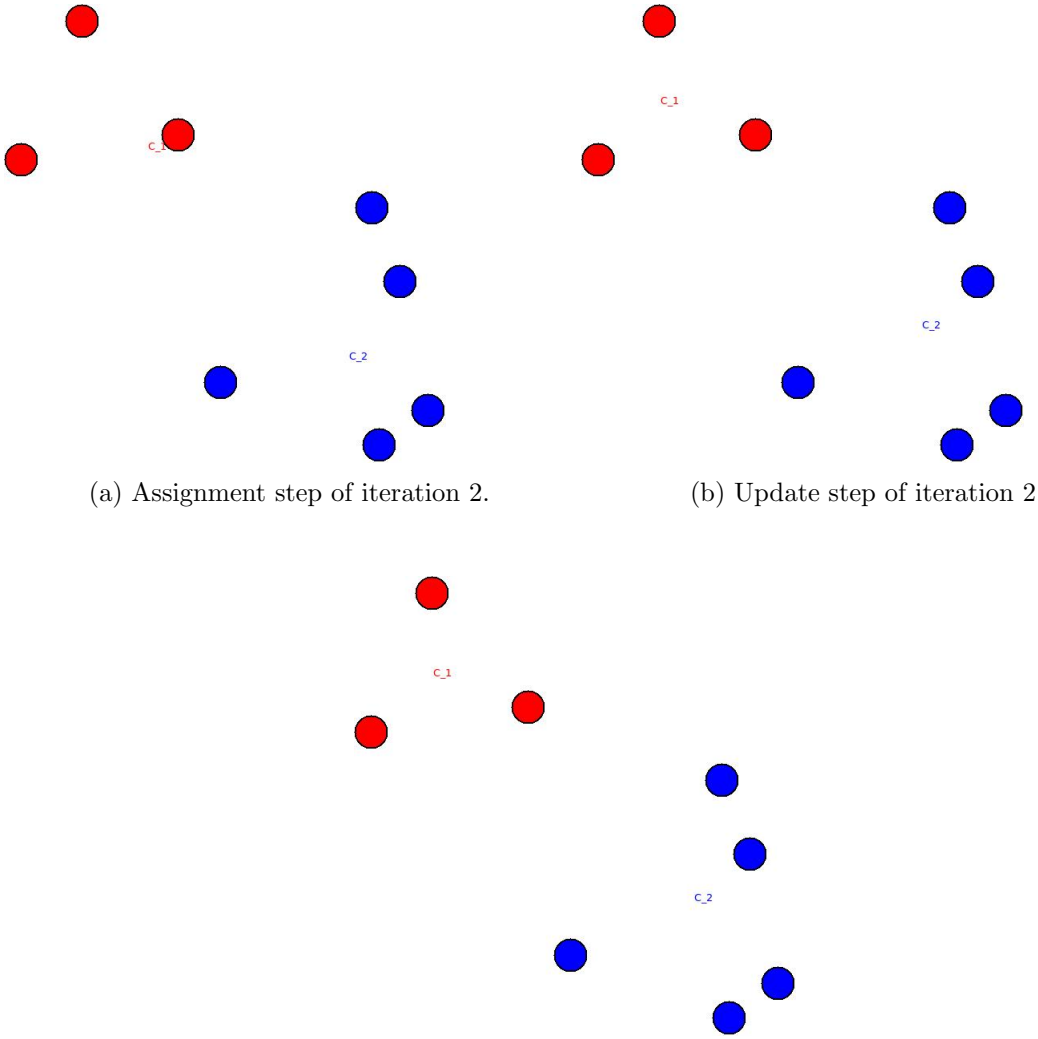
(b) Initialization step using the Forgy method.



(a) Assignment step of iteration 1.



(b) Update step of iteration 1.



(a) Assignment step of iteration 2.

(b) Update step of iteration 2.

Figure 3.4.4: Centroids do not change in posterior iterations. Convergence.

Chapter 4

Experiments

This chapter is divided into five sections. First, a detailed description is given in 4.1 of the different experiments carried out, explaining which are the datasets analyzed and the models employed in each one of them. In Section 4.2, metrics used to measure the accuracy of the error intervals built are described. Implementation details of the code needed to perform these experiments can be found in 4.3. Section 4.4 presents an analysis of the results obtained in each experiment. Tables containing these results can be found in Section 4.5.

4.1 Experiments Description

4.1.1 Artificial Data

The goal of the first experiment is to test the usefulness of the fixed intervals proposed as a detector of noise in the input data, to see if these intervals could be a good tool to choose what noise distribution should be used in general noise SVR models, as the ones described in Section 2.4. To this purpose, we create an artificial dataset with targets as follows:

$$y_i = 2\cos x_i + 3\sin(2x_i) \ , \tag{4.1.1}$$

where x_i are uniformly distributed over $[0, 2\pi)$.

These targets are then corrupted with noise of four forms:

1. **Laplace** noise with zero mean, μ, and unit scale, σ.

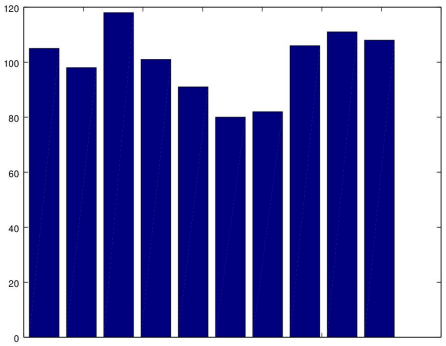
2. **Gaussian** noise with zero mean, μ, and unit variance, σ.

3. **Beta** noise with $\alpha = 1$ and $\beta = 2$.

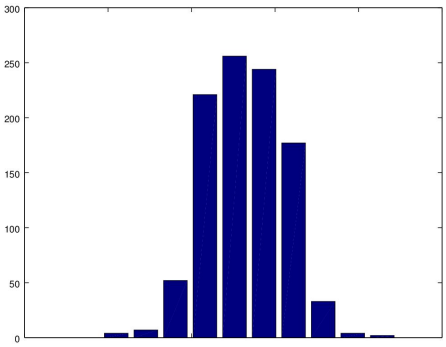
4. **Weibull** noise with unit scale, λ, and shape, κ, equal to 5.

A total of 5000 (x_i, y_i) pairs are generated. 4000 are used as train set and 1000 as test set. Then for each of these four corrupted artifical datasets, a SVR model is trained applying 5-fold cross-validation and out-of-sample residuals ψ_i are obtained. SVR parameters used are the ones obtaining lower CV error through a grid search. The details of this grid search are given in 4.3. The resultant residuals are used after that to generate the different prediction error intervals and their results are compared to see which ones yields better accuracy. Finally, we check if the distribution that gives the best results is the same as the one that the test described in 3.3 indicates as the best suited to the data.

As an example, the histograms of errors for the original artificial test set and the one corrupted with Laplace noise are shown in 4.1.1a and Figure 4.1.1b respectively. It is clear that a Laplace distribution fits better the latter distribution and, therefore, it is to be expected that the interval corresponding to this distribution is the one that achieves greatest accuracy. Results in 4.5 will confirm this hypothesis.



(a) Histogram of test errors for data without noise.



(b) Histogram of test errors for data corrupted with Laplace noise.

Figure 4.1.1: Errors for data without noise vs Errors for data corrupted with Laplace noise.

In this experiment and the next ones, we test the different intervals explained in Section 3.2. In addition, in [32] the authors show that in some cases for the Laplace interval is better to discard the extreme ψ_i values. The ψ_i we classify as extremes are the ones that exceed $\pm MS$, where M ranges from 3 to 5 depending on the problem and S is the standard deviation of the distribution of errors. We will refer to this as LAP* interval from now on. Thus, we test the following intervals:

1. **LAP:** Zero mean Laplace interval.
2. **LAP*:** Zero mean Laplace interval discarding extreme ψ_i values.
3. **LAPm:** General Laplace interval.
4. **GAU:** Zero-mean Gaussian interval.
5. **GAUm:** General Gaussian interval.
6. **BET:** Beta interval.
7. **WEI:** Weibull interval.

4.1.2 Public Datasets

The following public datasets are analyzed in this experiment:

1. **abalone:** From the Statlog collection [46]. 3177 instances used as train set and 1000 as test.
2. **space_ga:** From the StatLib collection [47]. 2107 instances used as train set and 1000 as test.
3. **add10:** From the Delve archive [48]. 8792 instances used as train set and 1000 as test.
4. **cpusmall:** From the Delve archive [48]. 7192 instances used as train set and 1000 as test.

Using these datasets the accuracy of three different approaches to build prediction error intervals are compared:

1. Fixed intervals as proposed in Section 3.2 using as SVR parameters the ones obtained by grid search using CV.
2. Fixed intervals as proposed in Section 3.2 using as SVR parameters the ones obtained by the Bayesian approach described in Section 2.3.
3. Bayesian intervals result of solving (3.1.1).

4.1.3 Wind Energy

The next experiment has as goal to apply the different techniques proposed to a real problem of wind energy prediction.

To this purpose, we use day ahead numerical weather prediction, NWP, surface wind, temperature, pressure and 100 meter wind forecasts provided by the European Centre for Medium-Range Weather Forecasts, ECMWF, in its 0.25° resolution model. At this resolution, the NWP grid for the Iberian peninsula and nearby coastal area has 1,995 points. Currently NWP updates are given at most every 6 hours, with two of them being the most accurate: forecasts at base hours 00 and 12. Since the ECMWF forecasts are given at three hour intervals we interpolate them linearly to hourly values. The target to predict is the hourly total wind energy production over the Iberian peninsula.

We use as dataset NWP forecasts ranging from January 1st 2011 to October 31th 2013. In our experience for both wind and solar energy prediction the use of a fixed validation set yields better results than applying cross-validation over the training set, so we use data from January 1st 2011 to December 31th 2011 as train, January 1st 2012 to October 31th 2012 as validation set and January 1st 2013 to October 31th 2013 as test[1]. Sliding training

[1]As data from November and December 2013 were not at our disposal when we carried out this experiment, we opted to leave data from 2012 corresponding to these months out of the validation dataset.

and validation sets as proposed in [16], where to build a model for month m, we use as validation subset month $m-1$ and months $m-13$ to $m-2$ as the training set, would probably give better predictions, but it would also make the process more computationally complex and as the goal is not to give the best predictions but to compare the different probability intervals, this simpler approach was preferred.

Using these datasets, six different approaches to build prediction error intervals have been followed:

1. $\boldsymbol{M^C_{s=1}}$: Obtain residuals ψ_i for the validation set using as SVR parameters the ones obtained after a grid search using the fixed validation set previously described. Employing these residuals use method proposed in Section 3.2 to get a unique interval for all instances in the test set.

2. $\boldsymbol{M^C_{s=2}}$**:** Obtain residuals ψ_i as in $M^C_{s=1}$. Split instances of validation set into $s=2$ groups, each with the most equal number of instances possible. Instances (x_i^{val}, y_i^{val}) of the validation dataset are divided in ascending order of their predicted value, $\hat{f}(x_i^{val})$, i.e., group 1, V_1, has instances with the lowest predicted values above all instances in the validation set. Thus, we set

$$V_m = \{(x_i^{val}, y_i^{val}) | \hat{f}(x^{val}{}_i) \in (p_{m-1}, p_m]\}, m = 1...s \ , \tag{4.1.2}$$

where $\{p_j\}_{j=1}^s$ are the $\frac{100j}{s}th$ percentiles of $\{\hat{f}(x_i^{val})\}_{i=1}^N$ and $p_0 = 0$

After that, divide residuals ψ_i into $s=2$ groups, $\{R_m\}_{m=1}^s$, where

$$R_m = \{\psi_i | (x_i^{val}, y_i^{val}) \in V_m\}, m = 1...s \ . \tag{4.1.3}$$

Use method proposed in Section 3.2 to get $s=2$ different intervals, $\{I_m\}_{m=1}^s$, one corresponding to each R_m. Then divide test set into $s=2$ groups, T_m

$$T_m = \{(x_i^{test}, y_i^{test}) | \hat{f}(x_i^{test}) \in (p_{m-1}, p_m]\}, m = 1...k \ , \tag{4.1.4}$$

where $\{p_j\}_{j=0}^s$ are as in (4.1.2)

Finally, test I_m over T_m

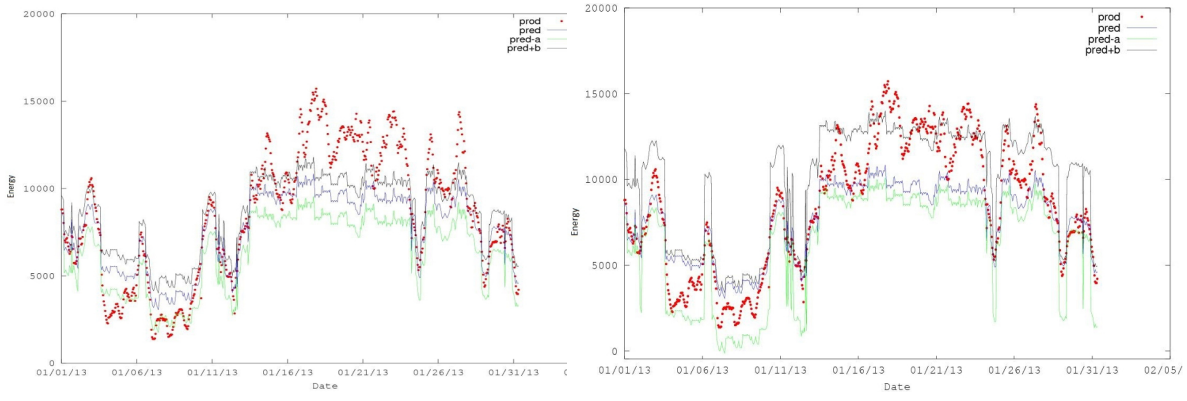
3. $\boldsymbol{M^C_{s=3}}$**:** Same method as in $M^C_{s=2}$ but this time with $s=3$

4. $\boldsymbol{M^C_{k=2}}$**:** Obtain residuals ψ_i for the validation set using as SVR parameters the ones obtained after a grid search using the fixed validation set previously described. Split instances of validation set into k groups, $\{V_m\}_{m=1}^k$, using k-means. Set $\{R_m\}_{m=1}^k$ and $\{I_m\}_{m=1}^k$ as in $M^C_{s=2}$. Divide test instances into k groups, $\{T_m\}_{m=1}^k$, assigning each instance (x_i^{test}, y_i^{test}) to T_m if centroid of cluster m is the closest to x_i^{test}. Test I_m over T_m. $k = 2$ is chosen because it yields the best results among values ranging from 2 to 10.

5. $\boldsymbol{M^B_C}$**:** Obtain residuals ψ_i for the validation set using as SVR parameters the ones obtained by the Bayesian approach described in Section 2.3. Employing these residuals use method proposed in Section 3.2 to get a unique interval for all instances in the test set.

6. $\boldsymbol{M^B_B}$**:** Compute uncertainty intervals using the Bayesian procedure in [28] in its entirety, i.e. intervals result of solving (3.1.1).

Notice that procedure $M^C_{s=1}$ yields a constant uncertainty interval for all input patterns x, while methods $M^C_{s=2}$ and $M^C_{s=3}$ yield either two or three intervals depending on the predicted value $\hat{f}(x_i)$. In principle, these intervals adjusted to the magnitude of energy forecasts are a rather sensible choice in wind energy predictions, as the prediction errors are very dependent on forecast values. A comparison between $M^C_{s=1}$ and $M^C_{s=2}$ intervals with $s = 0.1$ for January 2013 is shown in Figures 4.1.2a and 4.1.2b respectively. It is clearly observed that $M^C_{s=1}$ has a fixed width for all instances while for $M^C_{s=2}$ it varies between low and high values of $\hat{f}(x_i)$, allowing this interval to adjust better to the error curve.



(a) $M^C_{s=1}$ uncertainty intervals.

(b) $M^C_{s=2}$ uncertainty intervals.

Figure 4.1.2: $M^C_{s=1}$ and $M^C_{s=2}$ uncertainty intervals for January 2013 data of wind energy problem with $s = 0.1$. Figure shows real production (prod), prediction given by the model (pred), lower bound of prediction interval (pred-a) and upper bound (pred+b).

4.1.4 Solar Energy

This experiment is analogous to the one for wind energy prediction but now applied to the problem of solar energy prediction.

For this problem the NWP variables used as data are total cloud cover, TCC, and solar radiation directed downward at the surface, SSRD. These ECMWF variables are given as 3-hour accumulated values but this time no linear interpolation is employed. The reason for this is that in [16] models using disaggregated data yield worse results than the ones applied directly to the accumulated values. Thus, now the target to predict is the 3-hour accumulated solar energy production over the Iberian peninsula. Only hours ranging from 6 to 21, i.e. aggegated values 6, 9, 12, 15, 18 and 21, are considered.

We employ as dataset NWP ranging from December 1st 2012 to May 31th 2014. Data from December 1st 2012 to November 31th 2013 is used as train, December 1st 2013 to February 28th 2014 as validation set, and March 1st 2014 to May 31th 2014 as test. Again, sliding training and validation sets would probably give more accurate predictions but this simpler approach was preferred for the purpose of this paper.

Using these datasets, eight different approaches to build prediction error intervals have been followed. Most of them are equivalent to the ones detailed for the wind energy experiment but we added three models. The reason for this is that several studies show that in the particular case of solar energy prediction differentiating between different hours when building models is notably beneficial, so division of datasets similar to the ones proposed in [49] are applied:

1. $\boldsymbol{M^C_{s=1}}$.

2. $\boldsymbol{M^C_{s=2}}$.

3. $\boldsymbol{M^C_{s=3}}$.

4. $\boldsymbol{M^C_{k=3}}$**:** Analogous to $M^C_{k=2}$ but this time $k = 3$ yielded the best results among values ranging from 2 to 10.

5. $\boldsymbol{M^C_{tri1}}$**:** Obtain residuals ψ_i as in $M^C_{s=1}$. Split validation and test sets into 3 groups, $\{V_m\}^3_{m=1}$ and $\{T_m\}^3_{m=1}$ respectively, where group 1 corresponds to low radiation hours 6 and 21, group 2 to medium radiation hours 9 and 18 and group 3 to high radiation hours 12 and 15. Set $\{R_m\}^3_{m=1}$ and $\{I_m\}^3_{m=1}$ as in $M^C_{s=2}$ and test I_m over T_m.

6. $\boldsymbol{M^C_{tri2}}$**:** Same as M^C_{tri1} but with group 1 corresponding to morning hours 6 and 9, group 2 to middle day hours 12 and 15, and group 3 to night hours 18 and 21.

7. $\boldsymbol{M^B_C}$.

8. $\boldsymbol{M^B_B}$.

4.1.5 Solar Energy Using Clear Sky Smoothing

A particularly important problem in solar energy prediction is the great difference existing between the central hours of the day versus morning and night hours. This difficulty is aggravated by the assumption taken in the proposed approach that the prediction error interval is not influenced by the input values $\{x_i\}_{i=1}^{N}$, so the interval is constant for all instances of the test dataset. To try to soften these negative effects, in this experiment we compute a clear sky curve for each day of the year, and then we divide each 3-hour accumulated production by its corresponding clear sky value in order to smooth the production curve.

Following an approach similar to the one proposed in [49] or [50], to compute the clear sky value, $CS_{D,H}$, for a particular day, $D = d$, and hour, $H = h$, we simply calculate the maximum solar radiation, $SR_{D,h}$, for that hour h among the interval of days ranging from five days before d to five days after d, i.e.:

$$CS_{d,h} = max(SR_{D,h}), D \in [d-5, d+5] \ . \tag{4.1.5}$$

There are many other methods to compute clear sky curves [51], but this simpler approach has been chosen for its ease and because from our previous experience it gives similar, or even better, results for our purpose here.

Thus, the datasets used in this experiment for training, validation and test are exactly the same as the ones in the previous solar energy experiment described in 4.1.4, but with the target divided by its related clear sky values.

Using these datasets, we replicate the models tested in Subsection 4.1.4. This time models M_C^B and M_B^B are not analyzed because they yielded clearly worse results in the previous experiment and, furthermore, they do not assume independence between prediction error interval and input values $\{x_i\}_{i=1}^{N}$.

4.1.6 Solar Energy Using Three SVR Models

Until now, only one SVR model has been trained using all the points in the training dataset. However, as will be explained in 4.4.4 and 4.4.5, clearly the approach that yields the best results for the solar energy problem is M_{tri1}^C, i.e. splitting validation and test sets into 3 groups, where group 1 corresponds to hours 6 and 21, group 2 to hours 9 and 18 and group 3 to hours 12 and 15. Thus, it looks like a logical next step to try splitting the train dataset into three groups using the same methodology and then fit a different SVR model to each one of these groups of train points.

Datasets used for training, validation and test are exactly the same as the ones in 4.1.4.

For this experiment, only the model M_{tri1}^C is tested, as it is the one that yielded the best results in 4.1.4 and the one that is used to define how the train points are splitted and then used to train different SVR models.

4.1.7 Sporting Events Prediction

The prediction of sporting events outcome has been the subject of study in different works, such as [52] for the NFL[2] and [53] for the English Premier League[3]. Computing confidence intervals is particularly useful for this real-world problem, as it can be used for automatic decision-making regarding the placing of bets in sports betting systems or at least as a supporting tool for expert knowledge to make the decision of betting or not in a specific market. Here is essential to give probabilities to predictions, so we can choose to bet if our predicted probability for a particular outcome is bigger than the one given by the market. Furthermore, betting markets use this kind of techniques to determine its odds for a particular event.

In this experiment we focus our attention in the prediction of NBA[4] games outcome. To this purpose, we built a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ for the 2014-2015 regular season where each instance i corresponds to a specific game, there are $N = 1230$ games in the regular season, y_i is the number of points scored by the home team minus the points scored by the away team, and x_i is a vector with the following statistics:

1. Statistics regarding standings and number of victories for home and away team.
2. Statistics regarding home and away victories for home and away team.
3. Basic offensive and defensive statistics for home and away team.
4. Advanced offensive and defensive statistics for home and away team.
5. Whether the game went to overtime or not.

Information regarding point 2 was gathered from `http://www.nba.com`. Values 1, 3, 4 and 5 described before were collected from `http://www.basketball-reference.com`. Figure 4.1.3 shows an example of the data collected from the latter site. After removing redundant variables and those with low correlation with the target, we end with a total of 27 variables, most of them corresponding to group 4, i.e. advanced statistics.

Regular Season Glossary · SHARE · Embed · CSV · PRE · LINK · ?

Date		Visitor/Neutral	PTS	Home/Neutral	PTS		Notes
Tue, Oct 29, 2013	Box Score	Orlando Magic	87	Indiana Pacers	97		
Tue, Oct 29, 2013	Box Score	Los Angeles Clippers	103	Los Angeles Lakers	116		
Tue, Oct 29, 2013	Box Score	Chicago Bulls	95	Miami Heat	107		
Wed, Oct 30, 2013	Box Score	Brooklyn Nets	94	Cleveland Cavaliers	98		
Wed, Oct 30, 2013	Box Score	Atlanta Hawks	109	Dallas Mavericks	118		
Wed, Oct 30, 2013	Box Score	Washington Wizards	102	Detroit Pistons	113		
Wed, Oct 30, 2013	Box Score	Los Angeles Lakers	94	Golden State Warriors	125		
Wed, Oct 30, 2013	Box Score	Charlotte Bobcats	83	Houston Rockets	96		
Wed, Oct 30, 2013	Box Score	Orlando Magic	115	Minnesota Timberwolves	120	OT	
Wed, Oct 30, 2013	Box Score	Indiana Pacers	95	New Orleans Pelicans	90		
Wed, Oct 30, 2013	Box Score	Milwaukee Bucks	83	New York Knicks	90		
Wed, Oct 30, 2013	Box Score	Miami Heat	110	Philadelphia 76ers	114		
Wed, Oct 30, 2013	Box Score	Portland Trail Blazers	91	Phoenix Suns	104		
Wed, Oct 30, 2013	Box Score	Denver Nuggets	88	Sacramento Kings	90		
Wed, Oct 30, 2013	Box Score	Memphis Grizzlies	94	San Antonio Spurs	101		
Wed, Oct 30, 2013	Box Score	Boston Celtics	87	Toronto Raptors	93		
Wed, Oct 30, 2013	Box Score	Oklahoma City Thunder	101	Utah Jazz	98		
Thu, Oct 31, 2013	Box Score	New York Knicks	81	Chicago Bulls	82		

Figure 4.1.3: NBA games data in `http://www.basketball-reference.com`.

[2]National Football League, professional American football league.
[3]English professional league for men's association football clubs at the top of the English football league system.
[4]National Basketball League, pre-eminent men's professional basketball league in North America.

Values of x_i for each team correspond to the current ones by February 4th, where 728 games have been played. These games are used for training and validation, using 5-fold cross-validation. The remaining 502 games, for which no statistical data of the teams is included in x_i, are utilized for testing purposes.

For this experiment, we use models $M^C_{s=1}$, $M^C_{s=2}$, $M^C_{s=3}$ and $M^C_{k=2}$. Again, we do not consider models M^B_C and M^B_B because they produced clearly worse results in previous experiments.

4.1.8 Cancer Prediction

Our proposed approach is tested in this experiment on a dataset corresponding to an important real-world medical problem. Here we try to predict trachea, bronchus and lung cancer death rates country by country using past tobacco consumption data of their corresponding populations. This experiment has a double goal: First, to check the goodness of our method in a different kind of problem, one related to public health, and show its utility on this vital area. And second, to test how the proposed approach deals with the problem of having only a small sample of data to train the model and build the prediction error intervals.

Data for this problem is divided into two datasets:

1. **Train/Validation dataset:** 125 instances, each one corresponding to a different country. Input values, x_i, are the prevalence in 2000 of smoking any tobacco product among people aged 15 years or more[5], distinguishing between male and female individuals. Each target value, y_i, represents the sum of trachea, bronchus and lung cancer death rates in 2002 for a particular country.

2. **Test dataset:** Analogous to the train/validation dataset, but with smoking prevalence indicators of 2005 employed to predict 2008 trachea, bronchus and lung cancer death rate by country.

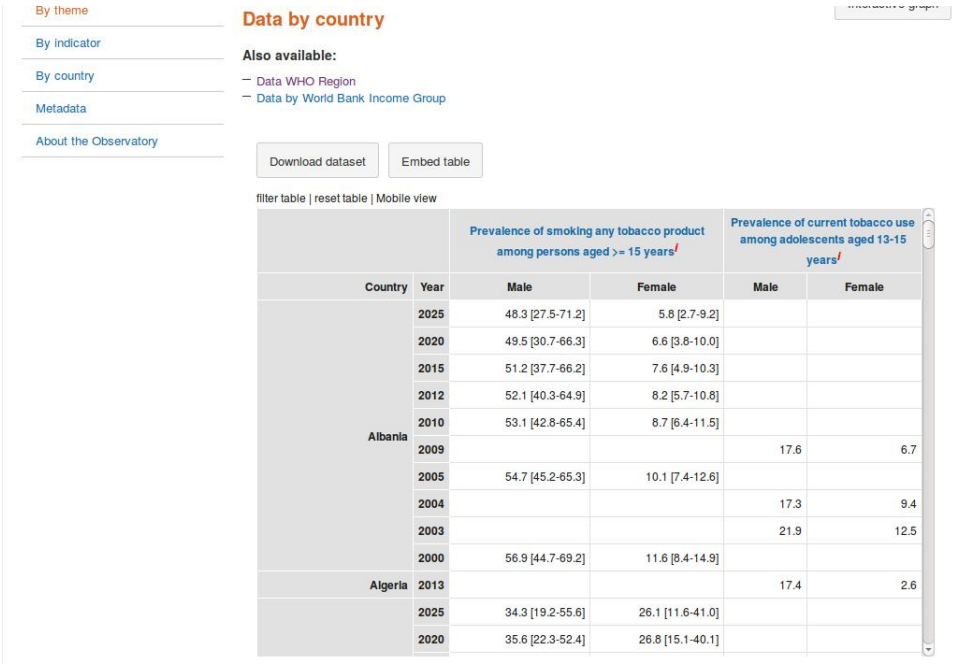
Thus, we have a dataset consisting of a 250x3 matrix, where 125 examples are used for train and validation through a 5-fold cross-validation approach, and 125 for testing. These datasets are collected from public data available at `http://www.who.int`[6]. Figure 4.1.4 shows an example of the data gathered from this site.

In this experiment, input vectors x_i only have two dimensions, one for the male population and other for the female population, so the suitability of using a SVR model for this problem instead of a simpler model like non-linear least squares regressor is not clear, but the former model has been chosen to keep consistency with previous experiments described in this work.

We test models $M^C_{s=1}$, $M^C_{s=2}$, and $M^C_{k=2}$. Results for models with greater values for s and k have not been shown because splitting into two groups already caused a decrease in accuracy, presumably because of the lack of enough data for this experiment, and results when dividing data into more groups were even worse.

[5]The smoking prevalence indicator estimates the age-standardized proportion of people age 15 years and older who are current smokers (daily or occasional cigarette smokers).

[6]World Health Organization web.



By theme
By indicator
By country
Metadata
About the Observatory

Data by country

Also available:
– Data WHO Region
– Data by World Bank Income Group

Download dataset | Embed table

filter table | reset table | Mobile view

		Prevalence of smoking any tobacco product among persons aged >= 15 years[1]		Prevalence of current tobacco use among adolescents aged 13-15 years[1]	
Country	Year	Male	Female	Male	Female
Albania	2025	48.3 [27.5-71.2]	5.8 [2.7-9.2]		
	2020	49.5 [30.7-66.3]	6.6 [3.8-10.0]		
	2015	51.2 [37.7-66.2]	7.6 [4.9-10.3]		
	2012	52.1 [40.3-64.9]	8.2 [5.7-10.8]		
	2010	53.1 [42.8-65.4]	8.7 [6.4-11.5]		
	2009			17.6	6.7
	2005	54.7 [45.2-65.3]	10.1 [7.4-12.6]		
	2004			17.3	9.4
	2003			21.9	12.5
	2000	56.9 [44.7-69.2]	11.6 [8.4-14.9]		
Algeria	2013			17.4	2.6
	2025	34.3 [19.2-55.6]	26.1 [11.6-41.0]		
	2020	35.6 [22.3-52.4]	26.8 [15.1-40.1]		

Figure 4.1.4: Smoking prevalence data in `http://www.who.int`.

4.2 Metrics

For cross validation and validation over a fixed set we use MAE

$$MAE = \sum \frac{|\hat{f}(x_i) - y_i|}{N} \ . \tag{4.2.1}$$

To test the error, err_s^M, of the error intervals estimated according to a certain noise model, M, that corresponds to a pre–specified probability, $1 - 2s$, we compare, as is done in [32], the percentage of the residuals δ_i^{test} lying in the estimated error interval I_s^M derived using M with the expected number, $(1 - 2s) \times N'$, with N' the test sample size, i.e.,

$$err_s^M = \frac{100}{N'} |\#\ of\ \delta_i^{test} \in I_s^M - (1 - 2s) \times N'| \ . \tag{4.2.2}$$

We choose an absolute error as accuracy measure over one with weights for positive or negative errors because preference towards a positive or negative error, i.e. which one is considered less detrimental of the two, is problem-dependent and here we opt to use a more universal measure.

4.3 Implementation Details

- For searching the best parameters by cross-validation or validation over a fixed set and then training and applying classical SVR, we use the software LIBSVM [38], available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. In our experience gaussian kernel yields the best results for wind and energy prediction problems, so we use it as kernel for the experiments.

Algorithm 2: Algorithm to select SVR parameters.

1 Set $(C, \epsilon, \gamma) = (C_0, \epsilon_0, \gamma_0)$ as the parameters obtained after using the *cherkassky's approach* [54].
2 **for** *i = 1 to nzooms* **do**
3 Select as grid points $[C_0 - \frac{5000}{2^i}, C_0, C_0 + \frac{5000}{2^i}] \times [\epsilon_0 - \frac{500}{2^i}, \epsilon_0, \epsilon_0 + \frac{500}{2^i}] \times [\gamma_0 - \frac{1}{2^i}, \gamma, \gamma + \frac{1}{2^i}]$, train SVR with each one of these points and apply models over validation set or using CV.
4 Set (C, ϵ, γ) as the grid point with lowest MAE over validation set or after CV.
end
5 Set final parameters as (C, ϵ, γ).

- To select the SVR parameters we compute a grid search using Algorithm 2. The value we use for *nzooms* depends on the dataset and the computational cost of training a SVR with it.

- For Bayesian Support Vector Regression, BSVR, the implementation from [28] is adopted.

- To implement the k-means algorithm we use scipy.cluster.vq package for python.

- For numerical integration, we use the SAGE function sage.gsl.integration.numerical_integral with its default values, i.e. adaptive integration and absolute and relative error tolerances equal to 10^{-6}.

- The rest of implementation needed, including code for the test described in 3.3, computing intervals detailed in Section 3.2, and splitting the data as described in 4.1.3, was developed for this paper.

4.4 Analysis

4.4.1 Artificial Data

In this experiment the goal is to study the usefulness of intervals proposed in Section 3.2 as a detector of noise distribution in the data. Results in Table 4.5.1 show that indeed for each of the four corrupted datasets the interval with the best results is the one corresponding to the distribution of the noise inserted in the data. Furthermore, the distribution corresponding to the best interval coincides with the one, or one of them, the test in 3.3 is in favour of, i.e. the one pointed out as the most likely distribution, even in the case of the data without noise. In fact, for some of the datasets, the test marks a pair of distributions as the most probable ones but finds it difficult to identify only one distribution of the two as the most probable with a low α value, while in the case of our proposed intervals the best choice of distribution has clearly the best results.

4.4.2 Public Datasets

Results of the different intervals proposed in Section 3.2 using CV parameters and the ones using parameters found by applying the BSVR approach explained in Section 2.3 are given in Table 4.5.2 and Table 4.5.3 respectively. Results for error bars computed by BSVR, i.e those coming from solving equation (3.1.1), are also shown in Table 4.5.3. A comparison of the best intervals for each case can be seen in Table 4.5.4.

The best accuracy is clearly achieved with the intervals result of computing SVR with CV parameters as the information in Table 4.5.4 shows. It is also interesting to note that indeed the new intervals added at the ones proposed in [32], detailed in Section 3.2, namely non-zero mean laplace and gaussian, beta and weibull intervals, are useful and yield good levels of accuracy, specially the first two. In fact, the overall winner, or one of them, for each of the four datasets belongs to this group of new proposed intervals.

4.4.3 Wind Energy

Several conclusions can be drawn from this experiment, whose results are summarized in Table 4.5.5:

- Accuracy of the proposed intervals in Section 3.2 is considerably high, clearly better than the one achieved with BSVR intervals.
- Non-zero mean Gaussian intervals produce the best overall results, although the Weibull distribution is the winner for some of the models proposed.
- Splitting data into two groups based on prediction values, $M^C_{s=2}$, causes a notably improvement in accuracy and is the overall winner of all models.
- On the contrary, dividing the dataset into 3 groups, $M^C_{s=3}$, prompts accuracy to decrease. Among the possible reasons for this effect are that there could be not enough data to create 3 consistent groups and that this approach forces a division into too many groups when there is not enough distinguishing common patterns in the data to do it.
- A separation using K-means, $M^C_{k=2}$, also enhances accuracy but less than $M^C_{s=2}$. It could be interesting to see if in the case of more data available, greater values of k could give even better results.
- CV parameters give higher accuracy than Bayesian parameters but intervals built following the approach described in Section 3.2 using Bayesian parameters are still preferable to BSVR intervals, i.e. the ones result of solving (3.1.1)

The extended table with accuracies of all combinations tested in this experiment can be found in Table B.0.11 of Appendix B.

4.4.4 Solar Energy

As in the wind energy problem, results of intervals proposed for this problem, shown in Table 4.5.6, yield notably high accuracy, even better than for wind energy prediction.

This time the most suited intervals for the data are clearly the ones corresponding to the zero-mean Laplace distribution, or its LAP* version, the winner for all the models proposed, with the only exception of $M^C_{s=2}$.

As also was the case in the wind energy experiment, CV parameters give the best results and again splitting the data into different groups is beneficial. However, this time a simple division based on prediction values is not enough to achieve the best results, and a more fine separation of the data is needed. K-means seems to be a good option but from the results obtained the conclusion is that the best one for this problem is to split the data taking into account the hour of the day corresponding to each instance of the dataset. Specifically, the best results are obtained for M^C_{tri1}, i.e., when the data is divided into 3 groups, one corresponding to hours 6 and 21, other to hours 9 and 18 and the last one to hours 12 and 15. This is consistent with what have been reported in previous experiments, as in [49], where this kind of division of the data gives the best prediction results for solar energy over the Iberian peninsula.

The extended table with accuracies of all combinations tested in this experiment can be found in Table B.0.12 of Appendix B.

4.4.5 Solar Energy Using Clear Sky Smoothing

As can be seen comparing results for this experiment, Table 4.5.7, versus results for the previous experiment, 4.5.6, using the clear sky smoothing on the target visibly improves accuracy of the model $M^C_{s=1}$ and, although lower, the rise in accuracy is also significant for $M^C_{k=3}$.

The remaining models, namely $M^C_{s=2}$, $M^C_{s=3}$, M^C_{tr1} and M^C_{tri2}, have much lower, almost unnoticeable, or non-existent improvements, and $M^C_{s=2}$ even suffers a slight decrease in accuracy for its best interval, the one corresponding to the Weibull distribution. The reason for this is probably that models $M^C_{s=2}$, $M^C_{s=3}$, M^C_{tr1} and M^C_{tri2} already take into account the difference between the various hours of the day by how they split validation and test datasets to build separate intervals for each group of test points.

As in experiment 4.1.4, the best model is again M^C_{tr1}.

The extended table with accuracies of all combinations tested in this experiment can be found in Table B.0.13 of Appendix B.

4.4.6 Solar Energy Using Three SVR Models

Results for this experiment are shown in Table 4.5.8. The use of three separate models, one used to fit hours 6 and 21, other to hours 9 and 18 and the third model to predict hours 12 and 15, instead of working with only one model for all points in the dataset, clearly provokes a rise in accuracy in the prediction error intervals.

This is an important result, as it suggests that an appropiate splitting of data and the use of different models for each cluster of points can significantly mitigate the negative effect of independence assumption between error interval and input vectors $\{x_i\}_{i=1}^N$ taken in the proposed approach, and this way achieve a notably improvement in the intervals accuracy.

These results agree with the ones presented in [49] and [50], where the suitability of splitting day hours and fitting separate models to each group of instances in order to obtain better solar energy predictions is shown.

4.4.7 Sporting Events Prediction

As shown in Table 4.5.9, accuracy of intervals built for these models, although still considerably high, is lower than for the previous experiments described. This could be explained by two factors: First, sporting events prediction have presented to be a difficult task in past studies. The human factor behind a sporting event outcome shows to be more hard to predict for a model than problems defined by a natural phenomenon dependant on underlying physics principles, as is the case for the wind and solar energy cases. Second, the target here is an integer in a delimited range, so errors are more condensed in a small area of values, which complicates the building of accurate error intervals.

This time, in contrast to what happened in the wind energy experiment, a simple division into two groups based on the magnitude of prediction values is not enough to obtain a significant improvement on accuracy. In contrast, splitting of data using the K-means algorithm yields great improvements in accuracy, appearing to better capture distinguishing patterns in the games. This may be explained because the correlation between predictions of high magnitude and high or low errors is not so clear here, as low magnitude targets corresponds to small differences between the number of points scored by each team and therefore to evenly matched games, more difficult to predict. They have small targets and potentially small errors, but normally the model will find more hard to predict these games so the relative error, $\frac{y_i - \hat{f}(x_i)}{y_i}$, will be higher. Moreover, division into three groups reduces accuracy, probably because of the lack of enough data available to create three separate clusters.

The best interval here shows to be the one corresponding to the Weibull distribution. It seems that a more extreme distribution like the Weibull better captures the underlying noise distribution for this problem.

The extended table with accuracies of all combinations tested in this experiment can be found in Table B.0.14 of Appendix B.

4.4.8 Cancer Prediction

Despite the small number of instances forming the dataset for this experiment, the proposed method still gives intervals with high levels of accuracy, as results in Table 4.5.10 show. Nonetheless, this lack of data appears to have a significant negative effect when splitting approaches are tested, as can be seen in the accuracy decrease for models $M^C_{s=2}$ and $M^C_{k=2}$, where it seems that more examples are needed to separate the dataset into self-consisting groups for error intervals building.

Lap* intervals give the best results. This may be caused again by the small sample of data available causing a few outliers to have a big impact on the overall accuracy results of the error intervals.

The extended table with accuracies of all combinations tested in this experiment can be found in Table B.0.15 of Appendix B.

4.5 Results

Table 4.5.1: Interval errors for the artificial data set with s=0.1.

Noise	Test	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI
Noise free	GAUm/WEI	3.1	3.1	3	1.5	1.4	2.5	**1.1**
Laplace	LAP/LAPm	**0.4**	**0.4**	**0.4**	1.4	1.4	1.0	1.5
Gaussian	GAUm/WEI	2.8	2.8	2.9	0.2	**0.1**	1.2	0.5
Beta	BET	3.0	3.0	3.0	1.1	1.2	**0.2**	0.8
Weibull	WEI	3.5	3.5	3.7	1.6	1.4	1.3	**0.2**

Table 4.5.4: Summary of interval errors and best noise models for public datasets.

Dataset		CV	BAYESIAN	BSVR
abalone	Best	LAP/WEI	LAP*	-
	Mean	**1.70**	3.55	4.95
add10	Best	LAPm	WEI	-
	Mean	**0.55**	1.40	4.00
space_ga	Best	GAUm	GAUm	-
	Mean	**1.35**	2.00	1.80
cpusmall	Best	LAPm	LAPm	-
	Mean	**0.20**	1.75	2.5

Table 4.5.2: Interval errors for SVR models with CV parameters for public datasets.

Dataset	*s*	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI
abalone	s=0.1	**2.8**	3.8	4.0	9.2	12	3.2	2.9
	s=0.05	0.6	**0.0**	0.8	1.9	0.8	1.6	0.5
	Mean	**1.7**	1.9	2.4	5.55	6.4	2.4	**1.7**
add10	s=0.1	0.4	0.4	**0.2**	0.6	0.8	0.4	3.0
	s=0.05	**0.9**	**0.9**	**0.9**	1.9	1.9	1.7	1.8
	Mean	0.65	0.65	**0.55**	1.25	1.35	1.05	2.4
space_ga	s=0.1	9.4	9.0	9.0	3.4	**2.2**	8.6	6.8
	s=0.05	3.0	3.9	2.0	0.7	**0.5**	3.9	4.3
	Mean	6.2	6.45	5.5	2.05	**1.35**	6.25	5.55
cpusmall	s=0.1	7.0	6.4	**0.2**	14.2	14.8	0.8	4.3
	s=0.05	4.1	3.9	**0.2**	6.9	6.5	**0.2**	2.7
	Mean	5.55	5.15	**0.2**	10.55	12.15	0.5	3.5

Table 4.5.3: Accuracy of intervals with Bayesian parameters for public datasets.

Dataset	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI	BSVR
abalone	s=0.1	**5.8**	6.2	12	13.6	9.8	6.8	**5.8**	7.0
	s=0.05	1.5	0.9	**0.7**	3.2	1.3	2.2	1.3	2.9
	Mean	3.65	**3.55**	6.35	8.4	5.55	4.5	**3.55**	4.95
add10	s=0.1	**1.0**	**1.0**	1.8	1.2	1.6	1.2	**1.0**	5.5
	s=0.05	1.9	1.9	**1.8**	2	**1.8**	2.3	**1.8**	2.5
	Mean	1.45	1.45	1.8	1.6	1.7	1.75	**1.4**	4.0
space_ga	s=0.1	9.7	9.3	9.3	3.9	2.5	8.1	7.5	**2.4**
	s=0.05	3.1	4.2	2.0	**1.2**	1.5	4.1	4.2	**1.2**
	Mean	6.4	6.75	5.65	2.55	2.0	6.1	5.85	**1.8**
cpusmall	s=0.1	6.5	6.5	**2.5**	11.2	11.7	2.8	5.6	3.8
	s=0.05	4.0	4.7	**1.0**	7.1	7.3	1.5	3.9	1.2
	Mean	5.25	5.6	**1.75**	8.65	9.5	2.15	4.75	2.5

Table 4.5.5: Summary of best interval errors for wind energy prediction.

	$M^C_{s=1}$	$M^C_{s=2}$	$M^C_{s=3}$	$M^C_{k=2}$	M^B_C	M^B_B
Best	WEI	GAUm	GAUm	GAU	WEI	-
s=0.1	4.5	**2.2**	7.0	3.6	5.5	6.9
s=0.05	3.2	**1.0**	5.2	3.7	4.4	5.7
Mean	3.85	**1.6**	6.1	3.65	4.95	6.3

Table 4.5.6: Summary of best interval errors for solar energy prediction.

	$M^C_{s=1}$	$M^C_{s=2}$	$M^C_{s=3}$	$M^C_{k=3}$	M^C_{tri1}	M^C_{tri2}	M^B_C	M^B_B
Best	LAP	WEIB	LAP	LAP	LAP	LAP*	LAP	-
s=0.1	1.5	2.0	2.5	1.4	**0.4**	0.9	3.3	6.4
s=0.05	**0.3**	1.3	0.8	**0.3**	**0.3**	0.8	0.8	3.6
Mean	0.9	1.65	1.65	0.85	**0.35**	0.85	2.05	5.0

Table 4.5.7: Summary of best interval errors for solar energy prediction using clear sky smoothing.

	$M^C_{s=1}$	$M^C_{s=2}$	$M^C_{s=3}$	$M^C_{k=3}$	M^C_{tri1}	M^C_{tri2}
Best	LAP	LAP*	LAP	LAP	LAP	LAP*
s=0.1	0.9	2.2	2.4	0.8	**0.4**	0.8
s=0.05	**0.3**	1.4	0.8	0.4	**0.3**	0.6
Mean	0.60	1.80	1.60	0.60	**0.35**	0.70

Table 4.5.8: Accuracy of intervals for solar energy with 1 model vs 3 models.

model	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI
1*model*	0.1	**0.4**	3.3	6.8	1.4	7.1	2.6	0.9
	0.05	**0.3**	1.9	3.2	0.8	6.1	3.1	4.2
	Mean	**0.35**	2.60	5.00	1.10	6.60	2.85	2.55
3*models*	0.1	**0.3**	0.8	1.3	0.7	7.0	2.7	0.9
	0.05	**0.2**	**0.2**	1.2	0.5	6.2	2.1	1.2
	Mean	**0.25**	0.50	1.25	0.60	6.60	2.40	1.05

Table 4.5.9: Accuracy of intervals for NBA prediction.

	$M^C_{s=1}$	$M^C_{s=2}$	$M^C_{s=3}$	$M^C_{k=2}$
Best	WEIB	WEIB	Gaum	WEIB
s=0.1	5.2	5.0	5.2	**2.6**
s=0.05	3.4	3.4	4.0	**1.4**
Mean	4.30	4.20	4.60	**2.00**

Table 4.5.10: Accuracy of intervals for cancer prediction.

	$M^C_{s=1}$	$M^C_{s=2}$	$M^C_{k=2}$
Best	LAP*	LAP*	LAPm
s=0.1	**1.6**	4.8	6.4
s=0.05	**0.8**	4.0	4.0
Mean	**1.20**	4.40	5.20

Chapter 5

Conclusions and Further Work

Support Vector Regression, SVR, is one of the most used tools in non–linear regression and modeling problems and, as such, uncertainty estimates of SVR predicted values are of great importance. A particularly clear example is, for example, SVR–based wind energy prediction, whose intermittency and wide fluctuations make necessary to define appropriate levels of rolling reserve; good uncertainty estimates are an obvious tool for this.

In this work, we have broadly followed the approach in [32], considering noise model distributions that are fitted to the residuals of SVR models whose C, ϵ and γ parameters are found directly by CV or validation over a fixed set, or under a Bayesian perspective; as in [32], we have also considered the full Bayesian approach to define uncertainty intervals proposed in [28]. We have enlarged this set up by adding to the noise models considered in [32] non–zero mean Gaussian and Laplace noise, as well as Beta and Weibull variants.

A first general conclusion is that, in agreement with [32], purely Bayesian interval estimates are poorer than those obtained by fitting noise distributions to residual values; moreover, interval error estimates are more accurate when SVR parameters are chosen by CV or validation over a fixed set. Since this is SVR specific only to the extent that SVRs are the underlying model, it suggests that direct residual–based fitting of error models should also be a useful approach when non–linear regressors are built under other alternative paradigms.

Intervals, detailed in Section 3.2, added at the two proposed in [32] produce considerably good levels of accuracy and their usefulness to give error prediction intervals seems to be notably high. Maybe other distributions, such as Cauchy distribution, Logistic distribution or Voigt distribution could also give good results for suitable datasets.

Moreover, we have shown over a synthetic example how this approach is able to resolve the true underlying noise model; this is the case for the four noise distributions considered, even when taking into account that the ϵ–insensitive SVR loss does not entirely corresponds to any of them. This mark them as a good tool to choose the noise assumption in a general noise SVR model. It would be interesting in future studies to verify if indeed a general noise SVR model assuming as distribution of data noise the one that gives better accuracy in the proposed intervals yields better results than if other distributions are chosen.

Of course, the true noise model depends entirely on the sample data and not, in principle, on the loss function used. On the other hand, the loss function somehow addresses a

particular noise structure, which suggests that perhaps loss functions other than the ϵ–insensitive one should be considered to build an SVR model to test if intervals that yield the best accuracy remain the same.

Non-zero mean Gaussian and Weibull distributions, depending on the model applied to build the intervals, give the best error prediction intervals for wind energy prediction and Laplace does it for solar energy. Weibull intervals are the best for the NBA games prediction problem and Laplace is the best choice for the medical data regression experiment. As stated before, checking if these distributions give the best predictions for these problems when used in a general noise SVR model could be a desirable idea for future research.

A drawback of the residual fitting approach is that error intervals are built independently of the $\hat{f}(x)$ regressor values. This is particularly so in problems such as wind energy prediction, where model errors are usually much higher for large energy values. Nevertheless, proposed intervals are shown to be a good option to build prediction intervals for problems such as wind and solar energy regression, specially when suitable splitting of the data is carried out. For wind energy a simple division based on the magnitude of prediction values is enough to achieve a great improvement on accuracy. However, this is still rather coarse and a finer grain split of the sample residuals into four regimes but using a single SVR model gives worse results. This also happens in the cancer prediction experiment, were the lack of data impose hard restrictions on the possibility of splitting the data. If there were more data available, it would be interesting to test if dividing the dataset into a larger number of groups produces even better results. For solar energy a more specific separation taking into account which hour of the day correspond to each instance is preferable. Applying the clear sky smoothing to the solar production curve shows to be other useful tool to reduce the impact of the independence assumption between error intervals and input vectors. Nonetheless, in other problems, such as sporting events prediction, a standard clustering algorithm, K-means, shows to be the best option for splitting the data.

It is also worth noting that the experiment performed where different SVR models are trained for each division of the data yields notably best results than when an unique SVR model is employed for all the dataset. This suggests that a good practice when using the proposed approach is to split the data, either by standard clustering techniques or by 'empirical' methods based on previous knowledge of the problem, and then train a different model for each one of the groups of points created.

The proposed approach gives notably good results when applied to a wide variety of real-world problems and dataset dimensions, ranging from solar and wind energy forecast to sporting events or medical predictions, suggesting that its use may be recommended for general purposes, and not only for problems belonging to the same topic or verifying some kind of common underlying characteristics.

The proposed technique for building error intervals is not exclusive for SVR approaches, with the method being independent of the model chosen to solve the regression problem and the noise assumptions presumed by this model. Thus, testing accuracy of intervals result of applying other regression models could be another line of further work.

We are currently working on these and other related issues.

Appendices

Appendix A

Appendix: Existence and Uniqueness of Solution to $G(\kappa) = \frac{\sum_{i=1}^{n} \log \psi_i}{n}$

Denoting $H(\kappa) = G(\kappa) - \frac{\sum_{i=1}^{n} \log \psi_i}{n}$ we want to prove the existence and uniqueness of the root of $H(\kappa) = 0$.

For $\kappa > 0$ and any $\psi_i \in \mathbb{R}^+$ the image of $H(\kappa)$ contains both positive and negative values and the function is continuous, so the existence of a root can be assured.

Now, to prove the uniqueness of this root we only need to show the global monotonicity of $H(\kappa)$ and this is equivalent to demonstrate $\frac{\partial H}{\partial \kappa} > 0$. The derivative takes the form:

$$\frac{\partial H}{\partial \kappa} = \frac{1}{\kappa^2} + \left(\sum_{i=1}^{n} \frac{1}{\psi_i^{\kappa}}\right)^2 \left(\sum_{i=1}^{n} \psi_i^{\kappa} \log^2 \psi_i \sum_{i=1}^{n} \psi_i^{k} - \left(\sum_{i=1}^{n} \psi_i^{\kappa} \log \psi_i\right)^2\right) \tag{A.0.1}$$

$\frac{1}{\kappa^2}$ and $\left(\sum_{i=1}^{n} \frac{1}{\psi_i^{\kappa}}\right)^2$ will always take positive values, so we focus on:

$$I(\kappa, n, \psi_i) = \sum_{i=1}^{n} \psi_i^{\kappa} \log^2 \psi_i \sum_{i=1}^{n} \psi_i^{k} - \left(\sum_{i=1}^{n} \psi_i^{\kappa} \log \psi_i\right)^2 \tag{A.0.2}$$

If $n = 1$:

$$I(\kappa, 1, \psi_i) = \psi_1^{\kappa} \psi_1^{k} \log^2 \psi_1 - \psi_1^{2\kappa} \log^2 \psi_1 = 0 \tag{A.0.3}$$

If $n = 2$:

$$I(\kappa, 2, \psi_i) = \psi_1^{\kappa} \psi_2^{\kappa} \left(\log \psi_2 - \log \psi_1\right)^2 \geq 0 \tag{A.0.4}$$

For $n \geq 3$, if we suppose $I(\kappa, n-1, \psi_i) \geq 0$ then:

$$I(\kappa, n, \psi_i) = I(\kappa, n-1, \psi_i) + \psi_n^{\kappa} \sum_{i=1}^{n-1} \psi_i^{\kappa} \left(\log \psi_n - \log \psi_i\right)^2 \geq 0 \tag{A.0.5}$$

This prove the global monotonicity of $H(\kappa)$ and consequently the uniqueness of the root of $H(\kappa) = 0$.

Appendix B

Appendix: Extended Tables

Table B.0.11: Accuracy of intervals for wind energy.

model	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI	BSVR
$M^C_{s=1}1$	0.1	6.0	5.9	5.0	5.6	5.5	5.0	**4.5**	-
	0.05	5.4	5.3	3.9	3.3	3.4	3.8	**3.2**	-
	Mean	5.7	5.6	4.45	4.45	4.45	4.4	**3.85**	-
$M^C_{s=2}$	0.1	5.1	4.8	2.4	3.2	**2.2**	4.7	2.6	-
	0.05	5.4	5.3	3.8	2.9	**1.0**	2.3	2.7	-
	Mean	5.25	5.05	3.1	3.05	**1.6**	3.5	2.65	-
$M^C_{s=3}$	0.1	8.5	8.1	7.2	7.6	**7.0**	8.3	7.8	-
	0.05	7.7	7.3	5.4	6.7	**5.2**	6.0	6.2	-
	Mean	8.1	7.7	6.3	7.15	**6.1**	7.15	7.0	-
$M^C_{k=2}$	0.1	6.2	5.7	5.2	**3.6**	5.6	4.4	4.0	-
	0.05	7.6	7.6	7.6	**3.7**	5.0	4.2	3.8	-
	Mean	6.9	6.65	6.4	**3.65**	5.3	4.3	3.9	-
M^B_C	0.1	8.2	8.1	7.8	6.1	5.8	6.9	**5.5**	-
	0.05	7.8	7.7	5.5	4.5	**4.3**	5.2	4.4	-
	Mean	8.0	7.9	6.65	5.3	5.05	6.05	**4.95**	-
M^B_B	0.1								**6.9**
	0.05								**5.7**
	Mean								**6.3**

Table B.0.12: Accuracy of intervals for solar energy.

model	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI	BSVR
$M^C_{s=1}$	0.1	**1.5**	2.0	4.1	4.3	4.2	4.3	3.2	-
	0.05	**0.3**	0.8	4.4	1.5	2.0	2.7	1.0	-
	Mean	**0.9**	1.4	4.25	2.9	3.1	3.5	2.1	-
$M^C_{s=2}$	0.1	2.1	**1.5**	4.4	5.1	5.8	5.3	2.0	-
	0.05	1.8	2.0	2.3	1.4	1.4	1.2	**1.3**	-
	Mean	2.0	1.75	3.35	3.25	3.6	3.25	**1.65**	-
$M^C_{s=3}$	0.1	**2.5**	2.7	5.1	8.2	7.0	7.3	4.1	-
	0.05	**0.8**	**0.8**	2.0	0.9	1.8	3.0	3.2	-
	Mean	**1.65**	1.75	3.55	4.55	4.4	5.15	3.65	-
$M^C_{k=3}$	0.1	**1.4**	2.2	4.7	1.7	3.1	2.6	3.1	-
	0.05	**0.3**	1.0	4.0	1.6	1.5	1.9	0.4	-
	Mean	**0.85**	1.6	4.35	1.65	2.3	2.25	1.75	-
M^C_{tri1}	0.1	**0.4**	3.3	6.8	1.4	7.1	2.6	0.9	-
	0.05	**0.3**	1.9	3.2	0.8	6.1	3.1	4.2	-
	Mean	**0.35**	2.6	5.0	1.1	6.6	2.85	2.55	-
M^C_{tri2}	0.1	1.5	**0.9**	4.1	1.5	3.3	5.2	2.6	-
	0.05	**0.5**	0.8	2.0	1.2	1.5	1.8	1.0	-
	Mean	1.0	**0.85**	3.05	1.35	2.4	3.5	1.8	-
M^B_C	0.1	**3.3**	**3.3**	4.7	4.3	4.6	6.9	3.2	-
	0.05	**0.8**	1.0	5.6	3.0	3.7	3.7	1.2	-
	Mean	**2.05**	2.15	5.15	3.65	4.15	5.3	2.2	-
M^B_B	0.1								**6.4**
	0.05								**3.6**
	Mean								**5.0**

Table B.0.13: Accuracy of intervals for solar energy using clear sky smoothing.

model	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI
$M^C_{s=1}$	0.1	**0.9**	**0.9**	3.1	4.2	3.8	3.5	3.3
	0.05	**0.3**	0.8	2.1	1.8	1.7	2.3	2.1
	Mean	**0.60**	0.85	2.60	3.00	2.75	2.90	2.70
$M^C_{s=2}$	0.1	**2.1**	2.2	3.4	5.8	5.8	3.7	2.6
	0.05	1.9	1.4	2.0	2.4	2.0	1.5	**1.3**
	Mean	2.00	**1.80**	2.7	4.1	3.9	2.60	1.95
$M^C_{s=3}$	0.1	**2.4**	2.6	3.1	6.2	6.1	5.1	4.3
	0.05	**0.8**	**0.8**	1.7	1.9	2.1	2.7	2.0
	Mean	**1.60**	1.70	2.40	4.05	4.10	3.90	3.15
$M^C_{k=3}$	0.1	**0.8**	1.2	2.7	2.0	2.1	2.1	2.1
	0.05	**0.4**	0.8	1.9	2.1	2.0	1.9	1.4
	Mean	**0.60**	1.00	2.30	2.05	2.05	2.00	1.75
M^C_{tri1}	0.1	**0.4**	1.3	2.8	2.4	6.1	1.9	1.6
	0.05	**0.3**	1.1	2.2	1.0	3.1	2.2	1.1
	Mean	**0.35**	1.20	2.50	1.70	4.60	2.05	1.35
M^C_{tri2}	0.1	1.3	**0.8**	2.1	1.4	2.3	2.2	2.0
	0.05	**0.5**	0.6	2.0	1.6	1.7	1.5	0.9
	Mean	0.90	**0.70**	2.05	1.50	2.00	1.85	1.45

Table B.0.14: Accuracy of intervals for NBA prediction.

model	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI
$M^C_{s=1}$	0.1	6.6	6.6	6.2	5.8	5.4	5.6	**5.2**
	0.05	5.6	5.8	4.8	4.4	**3.4**	4.0	**3.4**
	Mean	6.10	6.20	5.50	5.10	4.40	4.80	**4.30**
$M^C_{s=2}$	0.1	6.2	6.6	6.0	5.4	5.4	5.2	**5.0**
	0.05	5.6	5.6	4.6	4.0	3.6	**3.4**	**3.4**
	Mean	5.90	6.10	5.30	4.80	4.50	4.30	**4.20**
$M^C_{s=3}$	0.1	6.4	6.2	6.8	5.8	**5.2**	6.4	6.0
	0.05	5.6	5.4	4.8	4.4	4.0	5.2	**3.6**
	Mean	6.00	5.80	5.80	5.10	**4.60**	5.80	4.80
$M^C_{k=2}$	0.1	4.2	4.2	3.8	3.0	2.8	3.0	**2.6**
	0.05	3.2	3.4	2.8	2.0	2.0	1.8	**1.4**
	Mean	3.70	3.80	3.30	2.50	2.40	2.40	**2.00**

Table B.0.15: Accuracy of intervals for cancer prediction.

model	s	LAP	LAP*	LAPm	GAU	GAUm	BET	WEI
$M^C_{s=1}$	0.1	2.4	**1.6**	3.2	4.8	4.0	4.0	5.6
	0.05	2.4	**0.8**	2.4	4.0	3.2	4.0	4.8
	Mean	2.40	**1.20**	2.80	4.40	3.60	4.00	5.20
$M^C_{s=2}$	0.1	5.6	**4.8**	**4.8**	6.4	5.6	**4.8**	7.2
	0.05	4.8	**4.0**	4.8	4.8	5.6	4.8	5.6
	Mean	5.20	**4.40**	4.80	5.60	5.60	4.80	6.40
$M^C_{s=3}$	0.1	8.0	8.0	**6.4**	8.8	8.0	7.2	9.6
	0.05	4.8	**3.2**	4.0	4.8	4.8	5.6	6.4
	Mean	6.40	5.60	**5.20**	6.80	6.40	6.40	8.00

Bibliography

[1] Jackson P.: Introduction to expert systems. Addison-Wesley Pub. Co., Reading, MA (1986)

[2] Mitchell T.M.: The discipline of machine learning. Carnegie Mellon University, School of Computer Science, Machine Learning Department (2006)

[3] Bishop C.M.: Pattern recognition and machine learning. Springer New York (2006)

[4] Duda R.O., Hart P.E. and Stork D.G.: Pattern classification. John Wiley & Sons (2012)

[5] Hastie T., Tibshirani R. and Friedman J.: The elements of statistical learning. Springer (2009)

[6] Hoerl A.E. and Kennard R.W.: Ridge regression: Biased estimation for nonorthogonal problems. In: Technometrics, pp. 55-67, vol. 12, Taylor & Francis Group (1970)

[7] Beyer, M.A. and Laney, D.: The importance of 'big data': a definition. In: Stamford, CT: Gartner (2012)

[8] De Mauro, A., Greco M. and Grimaldi, M.: What is Big Data? A Consensual Definition and a Review of Key Research Topics. In: 4th International Conference on Integrated Information, Madrid, pp. 2341-5048, vol. 10 (2014)

[9] Laney, D.: 3D data management: Controlling data volume, velocity and variety. In: META Group Research Note, pp. 70, vol. 6 (2001)

[10] Shvachko K., Kuang H., Radia S. et al.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1-10, IEEE (2010)

[11] Kornacker M. and Erickson J.: Cloudera Impala: Real Time Queries in Apache Hadoop, For Real. `http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real` (2012)

[12] Ingersoll G.: Introducing Apache Mahout. In: Scalable, commercial-friendly machine learning for building intelligent applications. IBM (2009)

[13] Zaharia M., Chowdhury M., Franklin M.J. et al.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, pp. 10-10 (2010)

[14] Yang, H., Chan, L., King, I.: Support vector machine regression for volatile stock market prediction. In: Intelligent Data Engineering and Automated Learning-IDEAL 2002, pp. 391-396, Springer (2002)

[15] Kramer, O., Gieseke, F.: Short-term wind energy forecasting using support vector regression. In: Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011, pp. 271-280, Springer (2011)

[16] Gala Y., Fernandez, A., Diaz J. and Dorronsoro J.R.: Support vector forecasting of solar radiation values. In: Hybrid Artificial Intelligent Systems, pp. 51–60, Springer (2013)

[17] Shawe-Taylor J., Cristianini N.: An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press (2000)

[18] Fletcher R.: Practical methods of optimization. John Wiley & Sons (2013).

[19] Statnikov A., Aliferis C.F. and Hardin D.P.: A Gentle Introduction to Support Vector Machines in Biomedicine: Theory and Methods. World Scientific (2011).

[20] Pontil, M., Mukherjee, S.,Girosi, F.: On the noise model of support vector machines regression. In: Algorithmic Learning Theory, pp. 316–324, Springer (2000).

[21] Scholkopf B., Smola A.J., Williamson R.C., et al.: New support vector algorithms. In: Neural computation, pp. 1207-1245,vol. 12, MIT Press (2000)

[22] Scholkopf B., Bartlett P.L., Smola A.J. et al.: Shrinking the tube: a new support vector regression algorithm. In: Advances in neural information processing systems, pp. 330-336, MIT; 1998 (1999)

[23] Chang C. and Lin C.: Training v-support vector classifiers: theory and algorithms. In: Neural computation, pp. 2119-2147, vol. 13 MIT Press (2001)

[24] Platt J. et al.: Fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods-support vector learning, vol. 3, Cambridge, MA (1999)

[25] Zhou X., Ma Y., Cheng Z. et al. : A simplification on SMO algorithm and its application in solving ε-SVR with non-positive Kernels. In: 2010 IEEE International Conference on Information and Automation (ICIA), pp. 878-883, IEEE (2010)

[26] Fan R., Chen P. and Lin C. : Working set selection using second order information for training support vector machines. In: The Journal of Machine Learning Research, pp. 1889-1918, vol. 6, JMLR.org (2005)

[27] Gao, J.B., Gunn, S.R., Harris, C.J., Brown, M.: A probabilistic framework for SVM regression and error bar estimation. In: Machine Learning, vol. 46, pp. 71–89, Springer (2002)

[28] Chu, W., Keerthi, S.S., Ong, C.J.: Bayesian support vector regression using a unified loss function. In: IEEE Transactions on Neural Networks, vol. 15, pp. 29–44, IEEE (2004)

[29] Bludszuweit, H., Domínguez-Navarro, J. A., Llombart, A.: Statistical analysis of wind power forecast error. In: IEEE Transactions on Power Systems, vol .23, pp. 983–991, IEEE (2008)

[30] Hu, Q., Zhang, S., Xie, Z. et al.: Noise model based ν-support vector regression with its application to short-term wind speed forecasting. In: Neural Networks, vol. 57, pp. 1–11, Elsevier (2014)

[31] Celik, A. N.: A statistical analysis of wind power density based on the Weibull and Rayleigh models at the southern region of Turkey. In: Renewable energy, vol. 29, pp. 593–604, Elsevier (2004)

[32] Lin C., Weng R.: Simple Probabilistic Predictions for Support Vector Regression. National Taiwan University, Taipei (2004)

[33] Schölkopf B. and Smola A.J.: Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press (2002)

[34] Collobert,R., Sinz, F., Weston J. et al.: Trading convexity for scalability. In: Proceedings of the 23rd international conference on Machine learning, pp. 201–208, ACM (2006)

[35] Smola, A.J. and Schölkopf, B.: On a kernel-based method for pattern recognition, regression, approximation, and operator inversion. In: Algorithmica, vol. 22, pp. 211–231, Springer (1998)

[36] Suykens, J. and Vandewalle, J.: Least squares support vector machine classifiers. In: Neural processing letters, vol. 9, pp. 293–300, Springer (1999)

[37] Basak, D., Pal, S., Patranabis, D.C.: Support vector regression. Neural Information Processing-Letters and Reviews 11, 203–224, (2007)

[38] Chang, C., Lin, C.: LIBSVM: a library for support vector machines. In: ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, pp. 27, ACM (2011)

[39] Fisher, R.A.: Theory of statistical estimation. In: Mathematical Proceedings of the Cambridge Philosophical Society, vol. 22, pp. 700–725, Cambridge Univ Press (1925)

[40] Gnanadesikan, R., Pinkham, R.S., Laura P. Hughes.: Maximum likelihood estimation of the parameters of the beta distribution from smallest order statistics. In: Technometrics, vol. 9, pp. 607–620, Taylor & Francis Group (1967)

[41] Lehmann E.L., Romano J.P.: Testing statistical hypotheses. Springer Science & Business Media (2006)

[42] Hartigan, J.A.: Clustering algorithms. Wiley (1975)

[43] Hartigan, J.A. and Wong M.A.: Algorithm AS 136: A k-means clustering algorithm. In: Applied statistics, pp. 100–108, JSTOR (1979)

[44] Hamerly, G. and Elkan C.: Alternatives to the k-means algorithm that find better clusterings. In: Proceedings of the eleventh international conference on Information and knowledge management, pp. 600–607, ACM (2002)

[45] Kodinariya, T.M. and Makwana P.R.: Review on determining number of Cluster in K-Means Clustering. In: International Journal (2013)

[46] UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets.html

[47] StatLib Datasets Archive http://lib.stat.cmu.edu/datasets/

[48] Delve Datasets, http://www.cs.utoronto.ca/ delve/data/datasets.html

[49] Fernandez, A., Gala Y. and Dorronsoro J.R.: Machine learning prediction of large area photovoltaic energy production. In: Data Analytics for Renewable Energy Integration, pp. 38–53, Springer (2014)

[50] Gala Y., Fernandez, A. and Dorronsoro J.R.: Machine learning prediction of global photovoltaic energy in spain. In: International Conference on Renewable Energies and Power Quality, pp. 278 (2014)

[51] Bird R. and Hulstrom R.L.: Simplified clear sky model for direct and diffuse insolation on horizontal surfaces. Solar Energy Research Inst., Golden, CO (USA) (1981)

[52] Sinha S., Dyer C., Gimpel K et al.: Predicting the NFL using Twitter. In: arXiv preprint arXiv:1310.6998 (2013)

[53] Rue H. and Salvesen O.: Prediction and retrospective analysis of soccer matches in a league. In: Journal of the Royal Statistical Society: Series D (The Statistician), vol. 49, pp. 399-418, Wiley Online Library (2000)

[54] Cherkassky, V., Ma, Y.: Practical selection of SVM parameters and noise estimation for SVM regression. In: Neural networks, vol. 17, pp. 113–126, Elsevier (2004)